Universidad Autónoma de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO

PROCESAMIENTO DEL LENGUAJE NATURAL PARA EL ANÁLISIS Y RECONOCIMIENTO DE OPINIONES EN ENTORNOS EDUCATIVOS

Álvaro Villén Peinado
Tutor: Ruth Cobos

22/06/2018

PROCESAMIENTO DEL LENGUAJE NATURAL PARA EL ANÁLISIS Y RECONOCIMIENTO DE OPINIONES EN ENTORNOS EDUCATIVOS

Autor: Álvaro Villén Peinado
Tutor: Ruth Cobos

Escuela Politécnica Superior
Universidad Autónoma de Madrid

22/06/2018

Agradecimientos

En primer lugar, agradecer a Ruth Cobos la oportunidad de desarrollar este TFG y por la ayuda y orientación recibida.

Gracias a mis compañeros y amigos, tanto a aquellos que han estado conmigo desde el principio, como a los que me he encontrado en el final. Todos ellos me han acompañado en este camino y me han servido de apoyo y motivación en todas las dificultades encontradas.

Y por encima de todo, gracias a mi familia, por dármelo todo y no pedirme nada. Por apoyarme en los momentos más duros y por las incontables lecciones que nunca olvidaré.

Abstract

Abstract —

The continuous advances in the field of Information and Communications Technology (ITC) has favored the use of new techniques in teaching and learning.

One of the most popular training methods in the recent years is the Massive Open Online Courses (MOOC). MOOCs are open courses where, thanks to different platforms, more than 800 universities offer courses from different areas of knowledge. One of the most remarkable aspects is that the content of these courses is shown to the students in audiovisual and textual format, allowing the students to follow the courses in an autonomous way.

The edX-CAS (Content Analyzer System) project consists of a Web application that will be used as a tool for textual content analysis in courses taught by professors of Universidad Autónoma de Madrid (UAM) using the edX platform. The system will provide general information during the course as well as information related with the field of Natural Language Processing (NLP), focusing the analysis on emotions recognition.

The goal of this Final Degree Project is to facilitate the tasks of the teaching staff when examining the status of the course and, this way, to create new content that improves the students experience. Furthermore the system offers a modular and expandable structure to facilitate the addition of new areas of study.

Key words — Microservices, NLP, Learning Analytics, Data Science, Massive Open Online Courses, indicators

Resumen

Resumen —

El continuo avance que se produce en el campo de las Tecnologías de la Información y de la Comunicación (TIC) ha propiciado el empleo de nuevas técnicas en la enseñanza y en el aprendizaje.

Uno de los métodos de formación que más popularidad ha ganado en los últimos años es la de los MOOC (en Inglés: Massive Open Online Courses). Los MOOCs son cursos abiertos donde, gracias a diversas plataformas, más de 800 universidades ofrecen cursos de diversas ramas del conocimiento. Uno de los aspectos más destacables es que el contenido de los cursos es mostrado a los estudiantes en formato audiovisual y textual, permitiendo a los alumnos seguir la trayectoria de manera autónoma.

El proyecto edX-CAS (Content Analyser System) consiste en una aplicación Web que servirá de herramienta para el análisis de contenido textual en cursos impartidos por los profesores de la Universidad Autónoma de Madrid (UAM) utilizando como plataforma edX. El sistema proporcionará información general del transcurso de los cursos además de información relativa con el campo del Procesamiento del Lenguaje Natural (en Inglés: Natural Language Processing, NLP) enfocando el análisis al reconocimiento de emociones.

El objetivo de este TFG (Trabajo de Fin de Grado) es facilitar la tarea del profesorado al examinar el estado del curso y de esta forma realizar contenido que mejore la experiencia de los estudiantes. Además, el sistema ofrece una estructura modulada y ampliable para facilitar la incorporación de nuevos focos de estudio.

Palabras clave — Microservicios, NLP, Analíticas de Aprendizaje, Ciencia de datos, Cursos Online Masivos y Abiertos, indicadores

Índice general

Índice de tablas

Índice de figuras

1

Introducción

En este primer capítulo se detalla la motivación previa para focalizar la temática del proyecto. Posteriormente se definirán los objetivos a alcanzar en el Trabajo de Fin de Grado junto con una explicación de la estructura del presente documento.

1.1. Motivación del Proyecto

Los MOOC son cursos ofrecidos por universidades y plataformas online que plantean un nuevo enfoque en el paradigma del aprendizaje. Son cursos gratuitos a distancia que introducen nuevos métodos pedagógicos [3] basados en vídeos, problemas interactivos o gamificación (trasladar la mecánica de los juegos al ámbito educativo [4]. Además, permiten elegir el modo de certificación, de modo que los estudiantes pueden elegir el ritmo de aprendizaje en el que seguir el temario. La plataforma sobre la que se va a llevar a cabo el estudio es edX, una plataforma online destinada a la exposición de MOOCS, y más específicamente sobre los cursos dirigidos por la Universidad Autónoma de Madrid. Esta plataforma proporciona a las entidades académicas subscritas los datos que recopila de los perfiles de los estudiantes.

Por otro lado, NLP es el área de la investigación que estudia la forma en la que el lenguaje natural, almacenado en un sistema informático, puede manipularse de forma que preserve ciertos aspectos del original [5]. Para la generación de diversos indicadores, se ha hecho uso de diferentes técnicas de NLP con el objetivo de extraer toda la información posible de ellos. Gracias a la información proporcionada por edX, usaremos estos datos para conocer la intervención de los estudiantes en los foros, calificaciones, información demográfica... Además, para completar el análisis, se ha llevado un proceso de recopilación del contenido ofrecido por el profesorado, es decir, los documentos de texto con el temario

teórico, y un proceso de extracción de los subtítulos de los vídeos.

El objetivo del trabajo es el de reflejar por medio de una aplicación web, tanto datos referentes al transcurso general del curso, como la información extraída del texto. De esta forma llevaremos a cabo un estudio sintáctico, semántico y emocional haciendo uso de las herramientas que permiten este análisis en español.

1.2. Alcance

El presente TFG tiene por objeto diseñar una aplicación web formada por cuadros de mando que ayude en la gestión de analíticas de contenido mediante la visualización de una serie de indicadores. Más específicamente, los objetivos del trabajo son:

- Analizar los datos anonimizados de los MOOC proporcionados por la plataforma edX.
- Realizar un estudio sobre el procesamiento del lenguaje natural y de las características de los MOOC.
- Realizar un estudio de las tecnologías actuales para la implementación de aplicaciones Web.
- Llevar a cabo un análisis de los requisitos funcionales y no funcionales que debe alcanzar la aplicación.
- Diseñar la aplicación. El diseño debe incluir:
  - Arquitectura elegida y sus componentes.
  - Modelo de almacenamiento de datos elegido.
  - Diagrama de clase de los diferentes componentes de la aplicación.
- Desarrollar la aplicación: El desarrollo debe incluir:
  - Almacenamiento de los datos generales relacionados con los cursos y los estudiantes inscritos en ellos, y generación y almacenamiento de distintos indicadores a partir de los paquetes de datos.
  - Extracción y almacenamientos de los subtítulos de los vídeos y de los documentos PDF y HTML utilizados a lo largo de los MOOCs.
  - Perfilar la opinión positiva, negativa o neutra, además de la polaridad y objetividad del contenido textual de los MOOCs almacenado en la aplicación.
  - Procesar el contenido textual para logra una representación sintáctica y semántica de las palabras.
  - Exportación y visualización de los indicadores y estadísticas calculadas.
  - Integración de los procesos especificados en un una aplicación Web.

- Realizar un proceso de pruebas de integración y evaluación.
- Realizar un análisis de la aplicación resultante y el conjunto de mejoras y de trabajos futuros que se podrían llevar a cabo a partir del proyecto.

1.3. Estructura del documento

En el primer capítulo se ha hecho una introducción al proyecto, se ha detallado la propuesta exponiendo las motivaciones y se han establecido los objetivos del trabajo.

En el segundo capítulo se expone el estudio del estado del arte haciendo una introducción a los conceptos del aprendizaje electrónico, el procesamiento computacional del lenguaje natural, el análisis de sentimientos, la lematización, la representación vectorial de palabras, la descripción de arquitecturas para la construcción de proyectos y un estudio sobre las tecnologías utilizadas.

En el tercer capítulo se presentan las funcionalidades y el análisis de requisitos de la aplicación.

En el cuarto capítulo se explica el diseño utilizado para desarrollar la aplicación.

En el quinto capítulo se explica el proceso de desarrollo e implementación de la aplicación.

En el sexto capítulo se exponen las pruebas que se han realizado para comprobar el funcionamiento de la aplicación y el alcance de los objetivos.

En el último capítulo, se presentan las conclusiones finales y las propuestas de trabajos futuros.

2

Estado del Arte

2.1. Análisis del contexto

En esta sección, se expondrá una introducción a los conceptos del aprendizaje electrónico y al uso de distintas técnicas para la extracción de información sobre el texto.

2.1.1. Aprendizaje Electrónico

El Aprendizaje Electrónico es una modalidad de formación a distancia que ha buscado desde su concepción una dualidad pedagógica y tecnológica. Ésta segunda se debe a que el proceso educativo se realiza a través de sistemas software, principalmente aplicaciones web [6]. En este contexto aparecen los MOOC (Massive Open Online Courses), cursos abiertos dirigidos a una gran cantidad de usuarios y que cumplen una serie de características [1]:

- Estructura organizativa: el contenido se muestra dividido en función de la temática y es acompañado de actividades, ejercicios y pruebas con el fín de evaluar y acreditar la comprensión de las explicaciones.
- Carácter masivo: no existe límite en el número de inscripciones a ningún curso.
- En línea: el curso es impartido a través de Internet.
- Abierto: gratuito, se trata de una iniciativa para acercar la educación a culquier parte del mundo.

Existen diferentes plataformas para alojar MOOCs, siendo las más destacadas edX [7], Coursera [8] y FutureLearn [9].

2.1.2. Procesamiento Computacional del Lenguaje Natural

Entre las diferentes maneras de definir qué es el lenguaje, se podría explicar cómo el medio que es utilizado de manera cotidiana para establecer una comunicación con nuestro entorno, como podría ser un lenguaje de programación, y el lenguaje natural es aquel que ha evolucionado con el fin de servir de medio de comunicación entre seres humanos. El lenguaje natural goza de un gran potencial expresivo y ha venido siendo perfeccionado hasta hacer posible el análisis de situaciones complejas convirtiéndose en una herramienta útil para el razonamiento. Una de las principales tareas de la Inteligencia Artificial junto con la lingüística consiste en la comprensión de los lenguajes naturales con el fin de lograr una representación formal de éstos [10]. El campo encargado de combinar estas dos ramas es el Procesamiento Computacional del Lenguaje Natural (PNL). Los sistemas PLN utilizan una arquitectura basada en niveles independientes para proporcionar una solución a las diferentes tareas necesarias para la comprensión del lenguaje:

- Nivel Fonético: establece una relación entre las palabras y el sonido que representan.
- Nivel Morfológico: representan la construcción de las palabras a partir de unas unidades de significado más pequeñas, es decir, de su morfema.
- Nivel Sintáctico: establece el papel estructural que es asignado a cada palabra dentro de una oración.
- Nivel Semántico: trata el significado dependiente e independiente de una palabra dentro de un contexto.
- Nivel Pragmático: estudia como el significado de una frase puede verse afectado por las frases anteriores.

2.1.3. Análisis de sentimientos

Antes de hablar del análisis de sentimientos, conviene definir qué es la minería de texto. La minería de texto es una aplicación de la minería de datos en la que, por medio de información no estructurada, es decir, que no se encuentra registrada en ninguna base de datos, busca el descubrimiento de conocimiento. [12]. Esta información puede estar representada en textos de distinto formato, como puede ser en páginas web, documentos Word, PDF o contenido textual de redes sociales.

El análisis de sentimientos es una extensión de la minería de texto que tiene como objetivo la extracción de información subjetiva de contenido generado por usuarios y, mediante aprendizaje supervisado, poder obtener un valor de positividad o negatividad sobre un texto determinado [13]. La minería de texto también tiene como objetivo, aparte

de obtener la polaridad de un texto, determinar la subjetividad de las palabras o frases en función del contexto en el que se halle [14].

2.1.4. Lematización

En el uso del lenguaje es común modificar las palabras para lograr distintos significados gramaticales o categóricos alterando el lexema (modificando el género, número, conjugación, etc.) [15]. Con el objetivo de evitar estas alteraciones existen dos procesos conocidos en inglés como 'lemmatization' y 'stemming'. El proceso de lematización busca, en función de la raíz de la palabra y del contexto, la forma que por convenio se acepta para representar todas las formas flexionadas de una misma palabra. De esta forma, logramos que palabras como diré, dije o dijo se reduzcan al 'lemma' decir. La lematización se diferencia de la troncalización en que esta segunda trunca la palabra obteniendo un 'stem' o 'raíz' sin tener que esta ser, necesariamente, una palabra real [16]. Usando la troncalización lograríamos que, a partir de palabras como estudios y estudiantes, obtuviéramos la raíz estud.

2.1.5. Representación Vectorial de Palabras

La hipótesis distribucional de Harris [17] plantea que las palabras que se repiten en contextos similares tendrán significados similares. Gracias a esta premisa y al empleo de técnicas de procesamiento del lenguaje natural, es posible generar una representación vectorial de las palabras. Estos algoritmos se basan en el empleo de técnicas de aprendizaje automático no supervisado que, habiendo recopilado una cantidad importante de texto, es capaz de llevar a cabo una representación vectorial de las palabras gracias a las cuales es posible extraer información semántica y sintáctica. El modelo más estandarizado es word2vec, propuesto por Mikolov [18]. Existen varias implementaciones de word2vec: en primer lugar, se encuentra la implementación de los algoritmos CBOW y SGNS hechas por Mikolov, aunque también existen implementaciones en diversos lenguajes de programación, como la desarrollada en Python por Gensim [19], o mllib codificada en Java [20].

2.2. Definición de Arquitecturas

2.2.1. Arquitectura Monolítica

La arquitectura tradicional más popular hasta el momento es la conocida como monolíta. Esta arquitectura se caracteriza por implementar un tipo de software en el que todos los módulos están empaquetados en un único programa y que, por lo tanto, ha de desarrollarse sobre una misma plataforma. El resultado obtenido con esto es un software en el que todos los componentes son dependientes [21]. Esta arquitectura ha traído consigo

una serie de dificultades al querer adaptar la aplicación a nuevas tecnologías, escalar la aplicación eficientemente y optimizar el tiempo de despliegue entre otras situaciones. [22]

2.2.2. Arquitectura basada en Microservicios

El enfoque de los microservicios es aquel en el que, alejándose de la arquitectura monolítica, se busca desarrollar una arquitectura en la que todos los componentes estén poco acoplados y donde cada uno funcione como un servicio autónomo que trabaja junto con los demás [23].

Se busca que cada componente se ejecute en su propio proceso y que tenga medios para comunicarse con el resto de servicios. Normalmente se lleva a cabo por medio de una interfaz de programación de aplicaciones (API) sobre HTTP [24].

Esta arquitectura resulta interesante en entornos de producción, ya que, determinando que todo componente es reemplazable, se facilita la extensión e inclusión de nuevos módulos proporcionando una alta tolerancia a fallos y una alta disponibilidad.

Topologías Microservicios

Existen diferentes formas en la se puede decidir implementar la topología del software dependiendo del tipo de servicio que queramos prestar, pero tres de las más populares son:

- Topología basada en API REST: basada en la descomposición de sitios web en servicios pequeños y autónomos a los cuales se accede mediante una interfaz basada en REST. Esta topología es útil cuando se pretende prestar servicios a través de una API.
- Topología basada en aplicaciones REST: útil cuando las solicitudes del cliente son recibidas por medio de una interfaz de usuario. En esta situación, la API se despliega por separado del resto de servicios y se comunica con ellos usando peticiones REST.
- Topología de mensajería centralizada: difiere de las anteriores en que, en lugar de usar peticiones REST para el acceso remoto, utiliza un gestor de mensajes centralizado ligero. Dos tecnologías que permiten llevar esto a cabo son ActiveMQ y HornetQ. [25]

REST

Se trata de una arquitectura para sistemas hipermedia que permite, por medio de estándares como HTTP, XML, MIME, etc., el acceso a recursos utilizando los cuatro métodos principales en comunicaciones HTTP (GET, PUT, POST y DELETE) a través de un identificador universal de recurso (URI). Éste enfoque fue propuesto por Roy Fielding,

el cuál quería crear un estilo arquitectónico que reflejase mejor las propiedades deseadas de una arquitectura web moderna [26].

Contenedores

Los contenedores son imágenes de sistemas operativos virtualizados comparables con las máquinas virtuales, pero diferenciándose en que las máquinas virtuales virtualizan el hardware del servidor, y los contenedores virtualizan el sistema operativo de un servidor [27]. Los contenedores se convierten en componentes aislados e inmutables que autocontienen todo lo necesario para ser ejecutados. Por lo tanto, no importa dónde se esté ejecutando, que el resultado ha de ser el mismo. Estos contenedores cumplen la finalidad de prestar un servicio, por lo que tienen que ser accesibles a través de algún tipo de API. Este concepto de contenedor proviene de los principios de Unix [24], donde por medio de elementos como namespaces y cgroups se buscaba el aislamiento de los recursos y el empaquetado de las aplicaciones y de sus dependencias. Con esto, obtendríamos módulos independientes y reemplazables, logrando que cualquiera de ellos pueda ser retirado y sustituido por otro componente que, aún implementado en otro lenguaje, siga siendo compatible.

2.3. Estudio de las tecnologías

2.3.1. Bases de datos

MongoDB

MongoDB es un Sistema de Gestión de Bases de Datos (SGDB) NoSQL basada en un modelo de documentos JSON. Estos documentos son de tipo clave-valor y se agrupan formando colecciones, elemento proporcional a lo que sería una relación en un SGDB relacional. [28][29]

MySQL

Se trata del Sistema de Gestión de Bases de Datos relacional más popular en la actualidad. Su arquitectura relacional permite que la información sea almacenada en una serie de tablas con campos predefinidos y que, por medio de relaciones entre las distintas tablas, sea posible un acceso eficiente a la información. [28]

2.3.2. Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y

automatización de Virtualización a nivel de sistema operativo. Su tecnología se basa en los contenedores de Linux (LXC), pero a diferencia de éstos, Docker provee de APIs y herramientas de línea de comandos que hacen de su creación, distribución y ejecución, una tarea más sencilla. Además, Docker es el único proveedor de contenedores en la nube, proceso que lleva a cabo a través de su servicio Docker Cloud.

Los Docker permiten una arquitectura basada en 'microservicios', modularizando las aplicaciones que conforman los servicios obteniendo fragmentos independientes, de responsabilidad única y que se comunican unos con otros para dar la totalidad del servicio al usuario.

2.3.3. Tratamiento de texto

NLTK, Natural Language Toolkit

La principal herramienta utilizada a lo largo del trabajo es NLTK (Natural Language Toolkit). Se trata de una biblioteca especializada en el Procesamiento del Lenguaje Natural. Se ha utilizado esta biblioteca porque además de estar implementada y de ser usada en Python, es de código libre y cuenta con una amplia comunidad tanto en su desarrollo como aportando información sobre su utilidad. Algunas de las principales aplicaciones de NLTK son [30]:

- Desambiguación del sentido de la palabra: consiste en desambiguar automáticamente las palabras en función del contexto, explotando el simple hecho de que las palabras cercanas tienen significados estrechamente relacionados.
- Resolución del pronombre: el objetivo es el de la identificación de los sujetos de los verbos de una oración.
- Generación de lenguaje de respuesta: consiste en la capacidad de un sistema de generar una respuesta adecuada ante una situación teniendo en cuenta el contexto en el que se produce.
- Traducción: busca proporcionar una traducción idiomática entre cualquier par de idiomas.

Análisis morfológico EAGLES

El analizador morfológico para el castellano utiliza un conjunto de etiquetas para representar la información morfológica de las palabras. Este conjunto de etiquetas se basa en las definiciones propuestas por el grupo EAGLES para la anotación morfosintáctica de léxicos y corpus para todas las lenguas europeas [31]. Contempla la inclusión de accidentes gramaticales de los lenguajes, por lo que es posible que haya palabras que en un contexto no sea capaz de analizar. El empleo de esta clasificación permite determinar la calidad gramatical y la variedad léxica que se está empleando. Los atributos, valores y códigos

que establece EAGLES dependen de la categoría de la palabra. A modo de ejemplo se muestra la tabla correspondiente a los adjetivos en la Figura 2.1. Con esta tabla y usando el adjetivo 'alegre' como ejemplo, obtendríamos la etiqueta 'AQ0CP0'. Esto nos indicaría que la palabra se trata de un adjetivo calificativo, de género común y en plural.

ADJETIVOS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Adjetivo	A
2	Tipo	Calificativo	Q
		Ordinal	O
3	Grado	Aumentativo	A
		Diminutivo	D
		Comparativo	C
		Superlativo	S
4	Género	Masculino	M
		Femenino	F
		Común	C
5	Número	Singular	S
		Plural	P
		Invariable	N
6	Función	-	0
		Participi	P

Figura 2.1: Table de correspondencia de adjetivos: EAGLES

Gensim (word2vec)

Gensim es una librería que permite llevar a cabo un modelo mediante aprendizaje no supervisado sobre la semántica de las palabras a partir de texto plano. Permite llevar a cabo extracción de temas, indexación de documentos y la detección de similitudes con grandes corpus de texto. [32] Word2Vec es una herramienta de Gensim que proporciona una solución eficiente a la representación de palabras en forma vectorial. Esta herramienta toma como entrada un corpus de texto, formando el conjunto de entrenamiento, y con él determinará la representación vectorial que corresponde a cada palabra. [33]

TextBlob

TextBlob es una herramienta que, utilizando la tecnología de procesamiento del lenguaje natural de NLTK como base, hace uso de un modelo entrenado con unas calificaciones ya preestablecidas y, a partir del modelo obtenido, clasifica el texto aplicando Naive Bayes. Esto lo hace asignando una probabilidad de positivo o negativo [34].

TextBlob es una biblioteca muy completa que ha sido entrenada usando una amplia colección de libros famosos, guiones de películas, redes sociales y blogs entre otras fuentes.

TextBlob también nos permite entrenar nuestros propios datasets usando algoritmos de clasificación como Support Vector Machine o Árboles de Decisión aunque ésta no es la intención del proyecto [34].

Otro de los problemas a los que TextBlob da solución es a la interpretación de la objetividad de un texto. La dificultad de esta tarea reside en la valoración de esta propiedad tomando una frase o un texto dentro de un contexto. Para resolver esta tarea se aplica la misma mecánica de entrenamiento que con el análisis de la polaridad.

VADER (Valence Aware Dictionary and Sentiment Reasoner)

Se trata de una herramienta de análisis de sentimientos, basada en reglas, que está específicamente adaptada para analizar los sentimientos expresados en entornos sociales. Esta librería muestra diversas puntuaciones de las cuales la más útil es 'compound'. Esta puntuación se obtiene sumando la puntuación de cada palabra en el conjunto léxico, se ajusta a las reglas definidas y se normaliza el valor entre -1 y 1 (progresando de más negativo a más positivo). Este valor se desglosa para obtener la probabilidad de que el texto sea positivo, negativo o neutro. [35]

PyPDF2

PyPDF2 es una librería escrita en Python destinada a la manipulación de ficheros PDF. Entre otras capacidades, PyPDF2 permite dividir, fusionar, recortar y transformar las páginas de los archivos PDF. También puede agregar y modificar datos personalizados, opciones de visualización y contraseñas a archivos PDF. Puede recuperar texto y metadatos de archivos PDF, así como combinar archivos completos. [36]

Wordcloud

Librería de Python utilizada para la generación de mapas de palabras. [37]

BeautifulSoup4

BeautifulSoup4 es una librería de Python que permite extraer información de ficheros XML y HTML. [38]

2.3.4. Lenguajes de desarrollo y librerías

Python

Python es un lenguaje de programación de alto nivel, de propósito general, orientado a objetos o scripting e independiente de la plataforma. Como características principales

encontramos que su sintaxis es clara y simple, es OpenSource, interpretado en lugar de compilado, y con una amplia colección de librerías.

Flask

Flask es un microframework para Python basado en Werkzeug y Jinja2 [7]. Werkzeug es una colección de utilidades para aplicaciones WSGI la cual incluye debugger, todas las utilidades necesarias para llevar a cabo el manejo de peticiones y respuestas HTTP, control de caché, manejo de cookies, subida de ficheros, y un potente sistema de enrutamiento de URLs. Jinja2 es un potente lenguaje para el diseño de plantillas HTML en Python.

Pandas

Pandas es un paquete de Python que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para que trabajar con datos 'relacionales' o 'etiquetados' sea fácil e intuitivo. [39]

PyGal

PyGal es una librería de Python que permite la creación de múltiples tipos de gráficos SVG (Gráficos Vectoriales Escalables) en formato XML. Cuenta con una amplia variedad de opciones de personalización y de interacción logrando una representación eficiente de la información. [40]

Youtube DL

Permite obtener información acerca de videos alojados en la plataforma de YouTube. Además de poder descargar los vídeos de la plataforma, permite tener acceso a la metainformación de éstos. [41]

Scikit-Learn

Scikit-Learn es una librería de Python formado por una amplia gama de algoritmos de aprendizaje automático tanto supervisados como no supervisados facilitando en gran medida el uso y aplicación de estos algoritmos. [42]

3

Funcionalidad y Análisis de requisitos

3.1. Funcionalidad y Servicios

Una vez hecho un estudio sobre el estado del arte, es momento de definir los requisitos funcionales y no funcionales del software a desarrollar.

La aplicación se ha implementado siguiendo una estructura basada en microservicios acorde a las características de las topologías basadas en aplicaciones REST (explicada en el apartado 2.2.2). Por organización y simplicidad, se han agrupado los servicios en tres módulos descritos en la Figura 3.1.

3.1.1. Módulo de Almacenamiento

Este módulo se encarga tanto del almacenamiento de toda la información disponible de los cursos MOOC, como de los datos generados a partir de ellos (la estructura de este módulo se explica en la sección 4.2.1). Para su implementación se ha utilizado los Sistemas de Gestión de Bases de Datos MongoDB y MySQL.

3.1.2. Módulo de Cálculo

Este módulo es el encargado de la limpieza y del procesamiento de los datos de los cursos. Durante esta fase se generarán diferentes indicadores en función del elemento de análisis sobre el que se va a llevar a cabo el estudio (los indicadores son explicados en mayor profundidad en la sección 4.2.2). Se ha hecho uso de las siguientes librerías

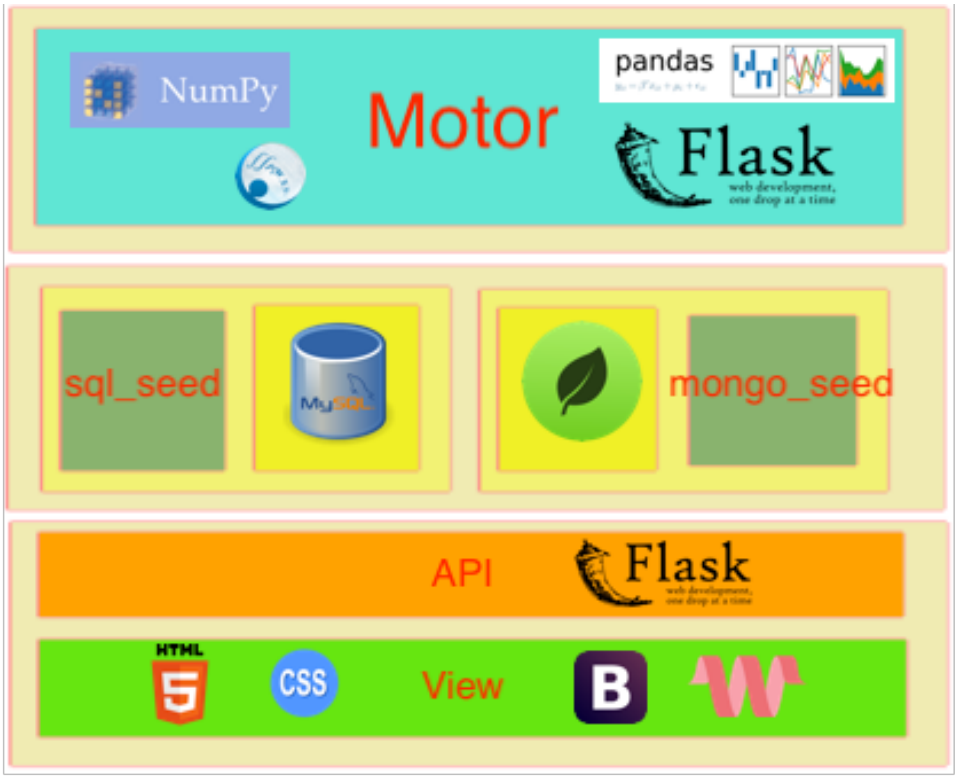

Figura 3.1: Diagrama de la Arquitectura de la aplicación

para la implementación de éste módulo: Flask, NLTK, Gensim, TextBlob, VADER, BeautifulSoup4, PyPDF2, Pandas, Youtube DL y Scikit-Learn. Los datos procesados provienen de las siguientes fuentes:

- Estudiantes: Los usuarios tienen la opción al registrarse de aportar información personal sobre ellos e indicar las metas que pretenden alcanzar al registrarse en la plataforma. Con estos campos obtenemos los términos más frecuentes, la diversidad léxica, la polaridad y subjetividad que representa a priori cada estudiante y una representación del análisis sintáctico de cada entrada siguiendo el formato definido en el apartado 2.3.3. Además de ello se procesa la información demográfica de los usuarios para obtener indicadores como la edad media o lugar de procedencia mayoritario.

- Certificados: Esta información no aporta contenido textual, por lo que su función es la de generar conocimiento sobre el transcurso del curso como puede ser la media de las calificaciones, el número de usuarios que han optado por la modalidad verificada y el número de usuarios activos.

- Vídeos: Los vídeos aportan contenido textual gracias a sus subtítulos, por lo que a partir de ellos analizamos la polaridad y subjetividad de éstos. También se recoge información sobre su duración, número de frases, tokens, positividad, negatividad y neutralidad, además de una representación vectorial de las palabras.

- Textos: Los dos componentes de texto que son tratados actualmente son PDF y HTML. Los textos contenidos en estos componentes son extraídos y procesados para tener un formato apto para su análisis consiguiendo extraer su polaridad y subjetividad, realizar un análisis sintáctico, una representación vectorial de las palabras y extraer la positividad, negatividad y neutralidad.

- Test: A lo largo del curso, el profesorado realiza una evaluación para comprobar el conocimiento adquirido por los estudiantes. Sobre estos test, se extraen las preguntas y respuestas con el objetivo de obtener su polaridad, subjetividad, positividad, negatividad y neutralidad y llevar a cabo un análisis sintáctico.

- Foros: A lo largo del curso, el profesorado propone temas que son discutidos en foros de debate. Esto aporta una fuente de conocimiento de donde se extraen indicadores como los votos que recibe cada comentario, su polaridad, subjetividad, positividad, negatividad y neutralidad y llevar a cabo un análisis sintáctico. Además, se mide la frecuencia de las palabras para obtener los términos más repetidos en las discusiones.

3.1.3. Módulo de Visualización

Este módulo se encarga de la representación visual de la aplicación y de la comunicación con el módulo de almacenamiento. Esto es posible gracias a dos servicios:

- API REST: representa una interfaz de programación que permite el acceso a los recursos almacenados. Para la implementación de este módulo se han usado las librerías Flask y Pandas.

- Interfaz Gráfica: se encarga de la generación de solicitudes a la API. Las librerías usadas en su implementación son Flask, Gensim, Pygal y BeautifulSoup4. También se ha hecho uso del conjunto de ficheros de estilo y scripts materializecss [27]. La interfaz se encarga de proporcionar los siguientes servicios:

    - Representar una interfaz gráfica para navegar entre los componentes de todos los cursos.
    - Representar una interfaz gráfica para la visualización de los indicadores generados para los componentes de vídeo y texto.
    - Representar una interfaz gráfica para la visualización de los diferentes objetos de estudio sobre los que se lleva a cabo el análisis (estudiantes, certificados, vídeos, textos, tests y foros).
    - Representar una interfaz gráfica para la exportación de los datos utilizados en las gráficas en formato csv con el fin de facilitar su posterior explotación y análisis.

3.2. Análisis de Requisitos

En las siguientes subsecciones se detallarán los requisitos funcionales y no funcionales generales del sistema.

3.2.1. Requisitos Funcionales

En este apartado se definirán tanto los requisitos funcionales propiamente como aquellos relacionados con la interfaz de usuario por su importancia en la aplicación, lo que hace que la mayoría de estos requisitos se puedan considerar como parte de la funcionalidad del sistema. Se presentan los requisitos en función de las fuentes de datos procesadas:

Estudiantes

- Req[01-01] El usuario podrá acceder a información estadística sobre los estudiantes.
  - El sistema detectará y procesará la información referente a los estudiantes de los cursos.
  - El sistema proporcionará información estadística sobre el número de usuarios almacenados en el sistema y el momento del curso en el que se inscribieron. Además, debe especificar el periodo de inicio y final del curso y la proporción de estudiantes en función del país de procedencia.
  - El sistema debe mostrar toda la información de forma granular, mostrando la información correspondiente a cada curso en específico o a modo comparativo entre ellos.
  - El sistema debe mostrar el número de estudiantes que han elegido la modalidad de curso verificado o si lo cursa como oyente, indicando cuántos de ellos están activos.
  - El sistema debe mostrar la edad de los estudiantes representada por cuartiles.
- Req [01-02] El usuario podrá analizar el contenido textual usado como descripción personal del estudiante.
  - A partir de la descripción que realiza cada estudiante al registrarse en la plataforma, el sistema debe representar la polaridad, subjetividad, positividad, negatividad, neutralidad, número de tokens, número de caracteres, diversidad léxica y las etiquetas del análisis sintáctico que corresponden a cada palabra.
- Req [01-03] El usuario podrá visualizar toda la información en español.
  - El sistema debe detectar si la información con la que el estudiante ha hecho su descripción es diferente al español, y en el caso de producirse esta situación ha de traducirlo.

- Req [01-04] El usuario podrá visualizar una nube de palabras.
    - El sistema debe lematizar las palabras de las descripciones de los usuarios y generar una nube de palabras adaptando el tamaño de las palabras a la frecuencia en la que aparecen.

Foros

- Req [02-01] El usuario podrá acceder a información estadística sobre los foros de debate.
    - El sistema detectará y procesará la información referente a los foros de debate de los cursos.
    - El sistema debe proporcionar información estadística sobre el número de votos positivos y negativos que reciben los comentarios en las discusiones.
    - El sistema debe mostrar toda la información de forma granular, mostrando la información correspondiente al foro de debate de cada curso en específico o a modo comparativo entre ellos.
- Req [02-02] El usuario podrá analizar los comentarios escritos por los estudiantes en los foros de debate.
    - El sistema debe mostrar información comparativa entre la polaridad, subjetividad, positividad, negatividad y neutralidad de los diferentes cursos.
    - El sistema debe mostrar la polaridad, subjetividad, positividad, negatividad, neutralidad, número de tokens, número de caracteres y las etiquetas del anális sintáctico que corresponden a cada intervención de un estudiante en un foro de debate.
- Req [02-03] El usuario podrá visualizar una nube de palabras.
    - El sistema debe lematizar las palabras de los comentarios de los estudiantes y generar una nube de palabras adaptando el tamaño de las palabras a la frecuencia en la que aparecen.

Videos

- Req [03-01] El usuario podrá visualizar el componente de vídeo que se pretende analizar.
    - El sistema debe detectar la sección en la que el usuario se encuentra ubicado y, en el caso de tratarse de un componente de video, buscará la URL que le corresponda e insertará un componente que permita su visualización.
- Req [03-02] El usuario podrá acceder a información estadística de los vídeos.

- El sistema debe detectar los vídeos presentes en los cursos y descargar todos los subtítulos vinculados.
- El sistema debe procesar los subtítulos debido a que las frases no suelen estar delimitadas por signos de puntuación.
- El sistema debe representar información estadística del contenido textual de estos subtítulos como el número de tokens, número de frases, el número de frases sin repetir y la duración del vídeo.
- El sistema debe ofrecer un análisis de la polaridad, subjetividad, positividad, negatividad, neutralidad, número de tokens, número de caracteres, diversidad léxica y las etiquetas del análisis sintáctico que corresponden a cada palabra para cada frase del texto.

Texto

- Req [04-01] El usuario podrá visualizar el componente de texto que se pretende analizar.
  - El sistema debe textar y procesar la información referente a los contenidos textuales de los cursos.
  - El sistema detectar la sección en la que el usuario se encuentra ubicado y, en el caso de tratarse de un componente de texto, buscará la entrada en la base de datos que le corresponda e insertará un componente que permita su visualización.
  - El sistema debe reconocer el tipo de componente de texto del que se trata, ya sea HTML o PDF.
- Req [04-02] El usuario podrá acceder a información estadística sobre el contenido de textual.
  - A partir de la de los componentes PDF o HTML recolectado de los cursos, el sistema debe representar información estadística del contenido textual de estos ficheros como el número de tokens, número de frases y el número de frases sin repetir.
  - El sistema debe ofrecer un análisis de la polaridad, subjetividad, positividad, negatividad, neutralidad, número de tokens, número de caracteres, diversidad léxica y las etiquetas del análisis sintáctico que corresponden a cada palabra para cada frase del texto.
- Req [04-03] El usuario podrá visualizar una nube de palabras.
  - El sistema debe lematizar las palabras de textos y generar una nube de palabras adaptando el tamaño de las palabras a la frecuencia en la que aparecen.

Tests

- Req [05-01] El usuario podrá acceder a información estadística sobre el contenido de textual de las preguntas realizadas en los test semanas.
    - El sistema deber proporcionar información estadística sobre el número de preguntas y el número de respuestas posibles planteadas en cada semana de cada curso.
- Req [05-02] El usuario podrá visualizar las preguntas y las posibles respuestas a dichas preguntas organizados por el número de la semana en la que se ha planteado.
    - A partir de los ficheros de textos recopilados con las preguntas realizadas en los tests, el sistema debe leerlos y organizarlos para ofrecer una interfaz en la que puedan ser visualizados.
    - El sistema debe ofrecer información como la polaridad, subjetividad, positividad, negatividad, neutralidad, número de tokens, número de caracteres, diversidad léxica y las etiquetas del análisis sintáctico que corresponden a cada palabra para cada frase de la pregunta y de las respuestas.

3.2.2. Requisitos No Funcionales

- Requisitos de rendimiento: El rendimiento del sistema ha de ser el suficiente para permitir una navegación fluida por la interfaz realizando un acceso eficiente al contenido almacenado. En cuanto a este aspecto, el rendimiento supone un reto debido a la gran cantidad de información que hay que almacenar y al tiempo de procesamiento que consume el cálculo de los indicadores.
- Restricciones de diseño: La aplicación se limitará a un formato Web, por lo que tendrá que ser un diseño responsive para que la aplicación pueda ser usada tanto desde dispositivos móviles como tabletas.
- Disponibilidad: La disponibilidad del servicio ha de ser absoluta a excepción de los momentos en los que sea necesario llevar a cabo labores de reparación o mantenimiento en el sistema, aunque se ha de lograr que estos tiempos sean mínimos.
- Seguridad: Los datos de los usuarios siempre irán encriptados para garantizar la seguridad de los datos.

4 Diseño

4.1. Arquitectura lógica

La arquitectura tradicional utilizada hasta ahora es la conocida como monolítica, donde todos los módulos están empaquetados en un solo programa y se implementa sobre una misma plataforma [21]. Uno de los objetivos de este TFG es el de crear una aplicación Web con una arquitectura descentralizada basada en microservicios. De esta forma, lograremos módulos de responsabilidad única que trabajen juntos [23]. Por la estructura de la aplicación, se propone una arquitectura que podría dividirse en tres módulos principales a los que denominaremos Capa de negocio, Capa de datos y Capa de presentación.

4.1.1. Capa de negocio

La capa de negocio es la encargada de la lectura, manipulación y generación de indicadores. Esta capa cumple dos tareas principales debido a la lógica de su implementación. Por un lado, se crea un proceso que lleva a cabo la carga de los datos almacenados en una estructura de directorios, la cual se explicará más adelante, y por otro lado se realizará el proceso de descarga de los subtítulos de los vídeos utilizados en los cursos.

4.1.2. Capa de datos

Esta capa se encarga del almacenamiento de toda la información analizada y recolectada en la capa de negocio. La capa de datos está compuesta de dos servicios principales: uno contiene la base de datos relacional MySQL y el otro la base de datos no relacional MongoDB (la información acumulada en ambas bases de datos es explicada en la sección 4.2.1).

4.1.3. Capa de presentación

Corresponde a la interfaz que sirve de interacción con el usuario. Es la encargada de mostrar los indicadores almacenados y calculados en las capas anteriores. Los controladores que gestionan la interfaz necesitan un medio para comunicarse con la Capa de Negocio, acción que se realizará por medio de una API REST.

4.2. Diseño por Servicios

Siguiendo la estructura basada en microprocesos, se han realizado contenedores para independizar las zonas de código más fácilmente reemplazables o ampliables. A lo largo de la sección se procederá a realizar una explicación detallada de la función que desempeña cada módulo del programa.

4.2.1. Módulo de almacenamiento

Base de datos MongoDB

EdX proporciona la estructura del curso incluyendo la relación entre los distintos elementos, componentes y enlaces de vídeos en un documento extraído de una base de datos MongoDB. En las siguientes ediciones se espera seguir recibiendo esta información del mismo modo, por lo que ha sido interesante mantener la tecnología y desarrollar un contendor para almacenar esta información. Esta base de datos se asume que es probable que sea ampliada con nuevos cursos o ediciones, para ello se ha implementado un script, process-course.py, para procesar los ficheros JSON donde los cursos almacenan la información descrita, y un contenedor configurado para cargar la salida del script en la base de datos MongoDB activa.

Base de datos MySQL

Por otro lado, el resto de la información es procesada en ficheros csv, por lo que convenía una estructura relacional que permitiese añadir la información extraída de

estos. Para la manipulación de los datos, se ha implementado una base de datos MySQL que sigue el mismo sistema de la base de datos MongoDB descrita, permitiendo recuperar copias en diferentes estados y facilitando así tanto su implementación como el mantenimiento. La estructura de la base de datos relacional es la que se muestra en la figura 4.1.

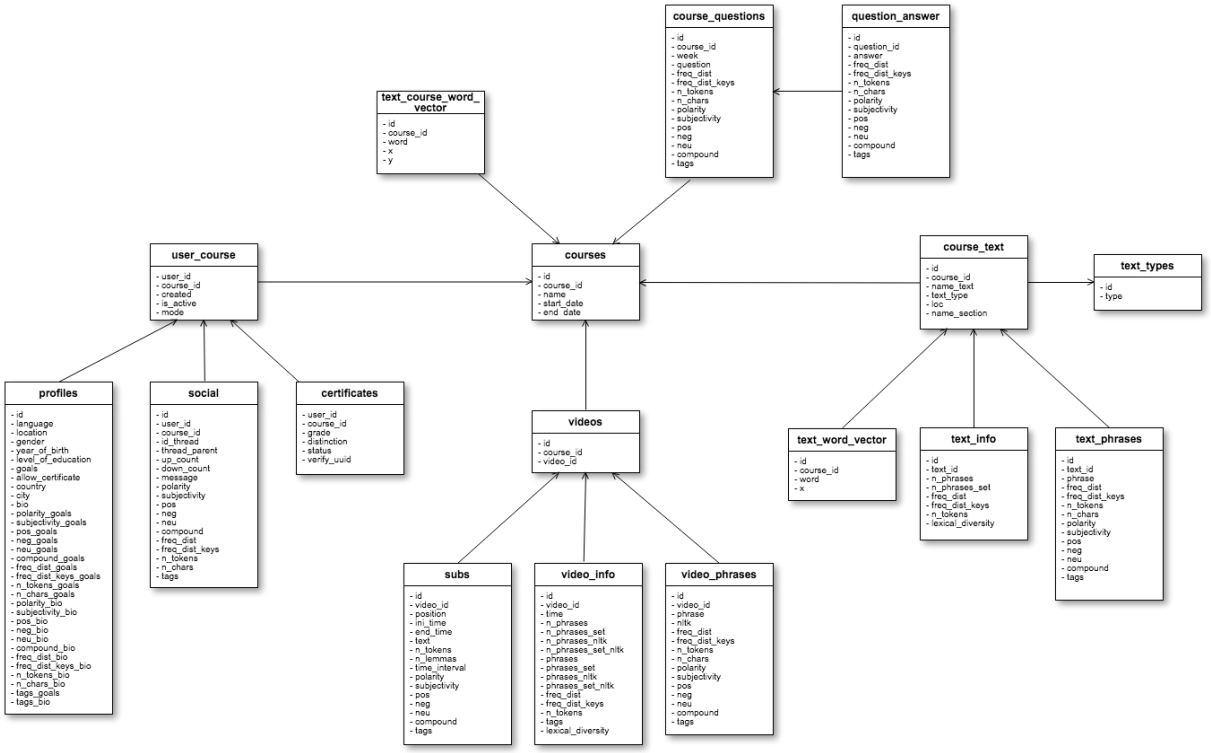

Figura 4.1: Diagrama Entidad Relación

- Tabla certificates: Esta tabla es la encargada de almacenar toda la información relacionada con los certificados de los estudiantes, como puede ser la calificación obtenida y el tipo de certificado elegido por el estudiante.
- Tabla text_types: Tabla auxiliar que permite distinguir el tipo de fichero que se ha manejado al extraer la información, actualmente HTML o PDF. Se decidió incluir esta tabla para facilitar la incorporación de posibles componentes futuros en otros formatos.
- Tabla course_text: Relación entre los textos analizados y el curso al que corresponden.
- Tabla course_questions: Almacena las preguntas de los test realizados a lo largo de cada curso.
- Tabla question_answer: Almacena las respuestas para cada pregunta de la tabla course_questions.

- Tabla courses: Contiene los identificadores a los datos de los cursos a analizar.
- Tabla text_course_word_vector: Usada para almacenar la representación vectorial de todas las palabras de cada curso.
- Tabla text_info: almacena diferentes indicadores generales de los textos de cada curso.
- Tabla text_phrases: Recolecta información de las frases de cada texto de los diferentes cursos.
- Tabla text_word_vector: Muestra una representación vectorial de cada texto de los cursos.
- Tabla profiles: Almacena información general de los estudiantes registrados en alguno de los cursos que están siendo sujetos de estudio.
- Tabla social: Contiene todas las intervenciones de los usuarios en los diferentes foros de debate.
- Tabla subs: Almacena todos los subtítulos junto con una serie de identificadores sobre ellos.
- Tabla user_course: Tabla que sirve de relación entre las tablas de profiles, social y certificates con la tabla de courses.
- Tabla video_info: Contiene la información general recolectada de los subtítulos de cada video.
- Tabla video_phrases: Contiene información relativa a cada frase de todos los vídeos de los cursos.
- Tabla videos: Tabla que relaciona los vídeos con el curso al que corresponde.

4.2.2. Módulo de Cálculo

Además de la carga de la información que nos proporciona edX, el sistema tiene que realizar tareas de una alta carga de ejecución para el procesamiento del texto. Conviene independizar esta tarea de la interfaz gráfica y de las bases de datos, ya que, en el caso de producirse un error, el fallo en el módulo no se propague al resto del sistema. La lógica de este proceso ha sido asignada a un contenedor, permitiendo así continuar interactuando con la interfaz mientras se realizan cálculos y recargas de información. La estructura usada para el módulo de cálculo es la que se muestra en la Figura 4.2.

Durante la recolección de los datos, el sistema calcula una serie de indicadores sobre los diferentes componentes de texto de los que se componen los curso. Estos componentes también son fraccionados en frases o segmentos de texto. Tanto sobre los componentes en su totalidad, como los segmentos de los que se compone, se generan una serie de indicadores que son explicados a continuación:

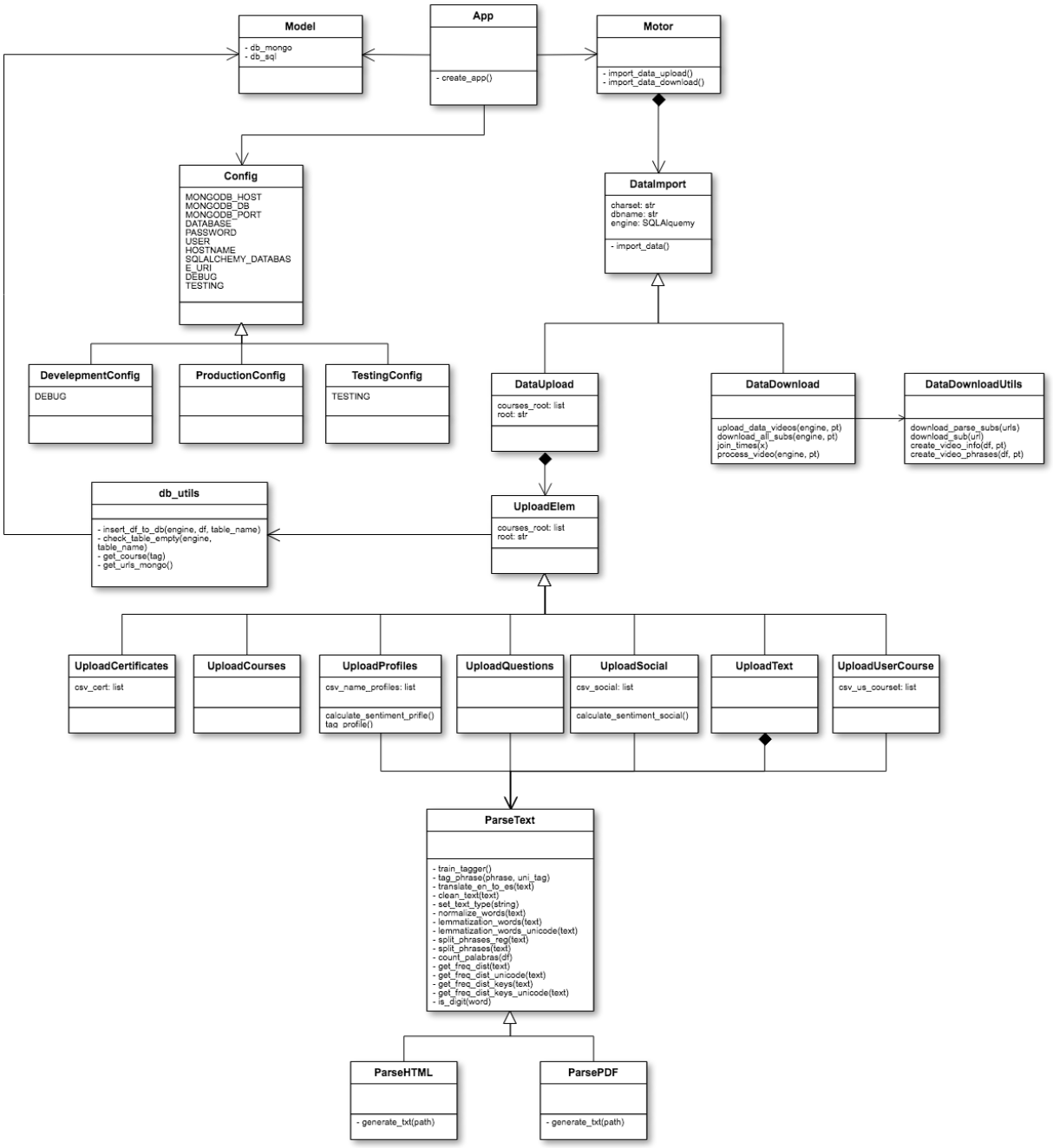

Figura 4.2: Diagrama de Clases

- Número de caracteres: proporciona la cantidad de caracteres que contiene un segmento o componente de texto.
- Número de tokens: dada una secuencia de palabras, se lleva a cabo un proceso de 'tokenización' logrando fraccionarlo en unidades mínimas denominadas 'tokens'. En este indicador se recoge el número de tokens de los que consta cualquier segmento o componente de un curso.
- Freciencia de distribución: este indicador aporta la frecuencia con la que, tras aplicar un proceso de lematización, aparece una palabra en un componente o segmento de texto.
- Número de frases: este indicador devuelve el número de frases que aparecen en un componente o segmento de texto.
- Número de frases diferentes: este indicador devuelve el número de frases diferentes que aparecen en un componente o segmento de texto.
- Diversidad léxica: este indicador refleja la variedad léxica de un texto como un factor entre el número de palabras sin repeticiones en un componente o segmento de texto entre el número de palabras totales. Para obtener un valor más real se lematiza el conjunto de palabras.
- Polaridad: este indicador, generado por la librería TextBlob, proporciona a un segmento de texto un valor de polaridad en el rango de [-1, 1] transcurriendo de más negativo a más positivo.
- Subjetividad: este indicador, también generado por la librería TextoBlob, asigna un valor de objetividad a una sección de texto en el rango de [0, 1] transcurriendo de más subjetivo a más objetivo.
- Positividad: este indicador, generado por la librería VADER, muestra la probabilidad de que un texto tenga carácter positivo.
- Negatividad: este indicador, generado por la librería VADER, muestra la probabilidad de que un texto tenga carácter negativo.
- Neutralidad: este indicador, generado por la librería VADER, muestra la probabilidad de que un texto tenga carácter neutro.
- Compuesto: este indicador, generado por la librería VADER, refleja el resultado agregado de la suma de las valencias calculadas de cada palabra, ajustada según una serie de reglas, y posteriormente normalizada en el rango de [-1, 1], transcurriendo de más negativo a más positivo.
- Etiquetas: este indicador devuelve el resultado de aplicar el análisis morfológico sobre cada una de las palabras de una frase.
- X, Y: estos dos indicadores reflejan la ubicación en el espacio que representa una palabra tras llevar a cabo el proceso de representación vectorial. El conjunto de palabras es previamente lematizado para reducir la dispersión de las palabras.

4.2.3. Módulo de Visualización

Se busca que cada módulo sea independiente de los demás, con la intención de que, si en un futuro se desease implementar o rediseñar alguno de ellos, exista el menor número de dependencias posible entre ellos. Por este motivo, para independizar los controladores de la base de datos, se ha implementado una sencilla API REST para acceder a la información almacenada en las bases de datos. Esta API será lanzada en un contenedor distinto al de la interfaz gráfica, y por medio de peticiones HTTP se mostrará la información solicitada sobre la interfaz.

API

La API desarrollada es de tipo REST. Se encarga de procesar las peticiones realizando consultas sobre la base de datos correspondiente y tramita la respuesta devolviendo un objeto JSON. Se muestra un diagrama del módulo en la figura 4.3.

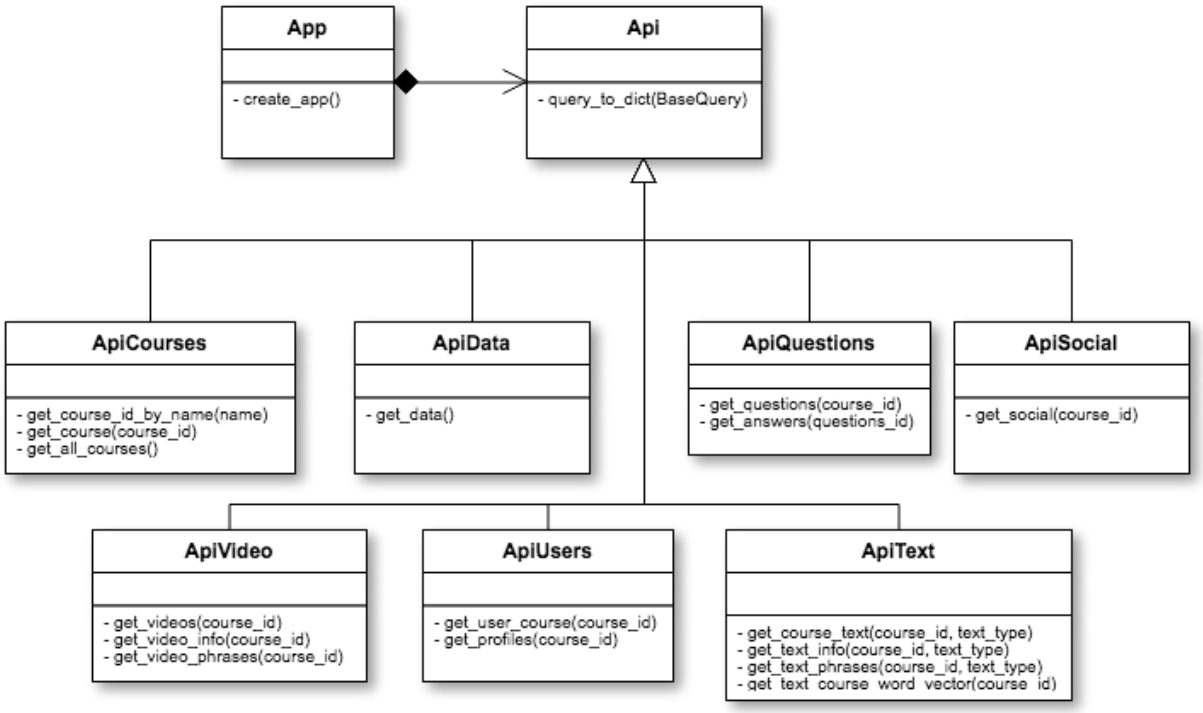

Figura 4.3: Diagrama de Clases: API

Interfaz gráfica

Este último módulo corresponde con el contenedor que contiene a la interfaz gráfica. Se encargará de realizar peticiones HTTP a la API, generará las gráficas e indicadores necesarios y los mostrará por pantalla.

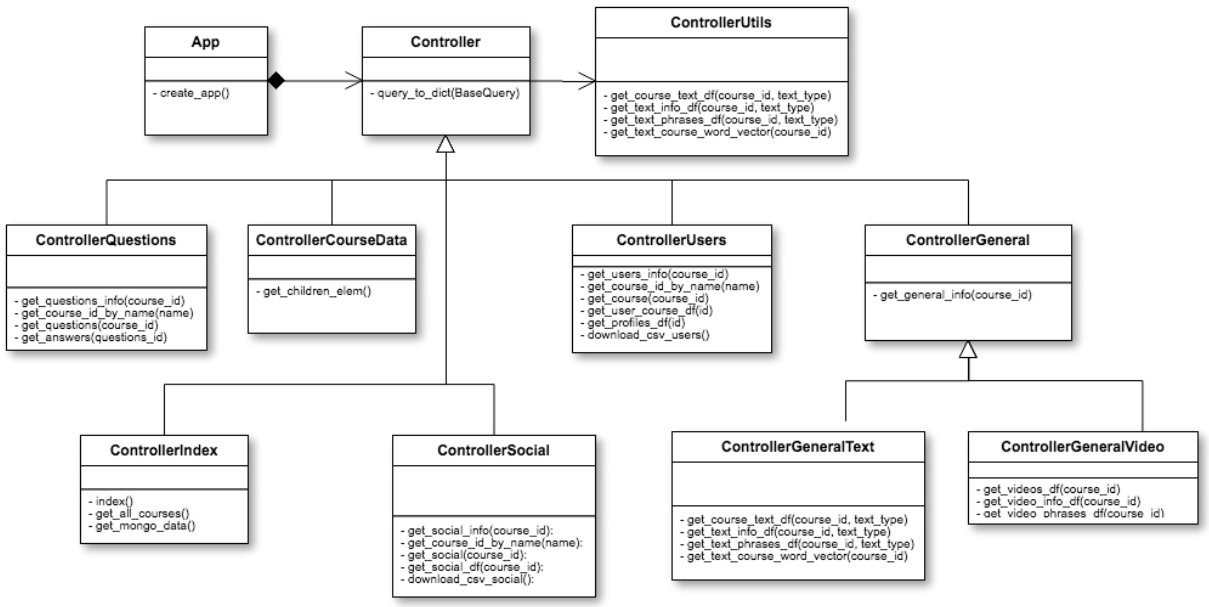

Figura 4.4: Diagrama de Clases: Vista

5 Desarrollo

5.1. Estructura de los ficheros

Como se ha especificado en la arquitectura del proyecto podríamos diferenciar cinco servicios principales. Se ha creado un directorio dedicado a cada uno de los servicios logrando de esta forma, y basándonos en la arquitectura explicada, que cada servicio autocontenga su propia sección de código. La estructura de directorios es la siguiente:

- api: este directorio contiene todos los ficheros necesarios para poner en funcionamiento una API REST que servirá de medio de comunicación entre cualquier servicio y las bases de datos. Al levantar el servicio se ejecutará el fichero index_api.py poniendo en funcionamiento un servidor implementado con el microframework Flask y, tras inicializarse, se mantendrá en modo de escucha hasta recibir una petición HTTP. En ese momento, accederá a los datos solicitados y serán devueltos en un fichero JSON. Los ficheros de los que se compone el directorio son los siguientes:
  - api.py: clase abstracta que gestiona la transformación de los tipos de datos recolectados de las bases de datos.
  - api_courses.py: implementación de la clase Api que permite obtener el identificador de un curso a partir de su nombre, obtener la información de un curso a partir de su identificador, y obtener toda la información de los cursos.
  - api_mongo.py: implementación de la clase Api para la gestión de la conexión con la base de datos mongo y devolver un diccionario con la estructura de los cursos.

- api_questions.py: implementación de la clase Api para la recolección de preguntas de test a partir del identificador del curso y de las respuestas en función del identificador de la pregunta.
- api_social.py: implementación de la clase Api para la recolección de las interacciones de los foros a partir del identificador del curso.
- api_text.py: implementación de la clase Api. Realiza consultas a la base de datos para obtener los textos de un curso en función del identificador de éste y del tipo de texto que sea, además permite obtener la información general de estos textos, las frases de las que están compuestos y acceso a la representación vectorial de las palabras.
- api_users.py: implementación de la clase abstracta Api para la gestión de los usuarios en función del curso al que pertenezcan.
- api_video.py: implementación de la clase Api para la obtención de los diferentes videos de los cursos, además de la información general de éstos y de las frases de las que están compuestos sus subtítulos.
- app.py: en este fichero se implementa la clase App, encargada de inicializar la aplicación, crear las conexiones necesarias con la base de datos, y de crear las rutas de acceso a los diferentes recursos descritos.
- config.py: fichero con las configuraciones necesarias para crear las conexiones necesarias con las bases de datos. Contiene una clase Config de la cuál heredan las clases ProductionConfig, DevelopmentConfig y TestingConfig. La instanciación de una u otra clase depende del entorno en el que se esté trabajando. Para el caso del proyecto es DevelopmentConfig.
- Dockerfile: fichero en el que se recoge la información necesaria para la creación del contenedor. Se hace uso de una imagen de Python 2.7.
- index_api.py: fichero de inicio del sistema. Instancia una clase App, asigna las configuraciones e inicia el servicio.
- model_mongo.py: clase que crea un modelo de la base de datos mongo. Recoge la información general de las distintas secciones del curso, especificando su categoría, su identificador, el identificador de las secciones hija a la actual y la metainformación asociada a dicha sección.
- model_sql.py: contiene las clases para la creación de los modelos de las distintas tablas de la base de datos SQL.
- requirements.txt: fichero en el que se especifican las librerías de python necesarias para la ejecución del programa.

- mongo: En este directorio se encuentran todos los ficheros necesarios para ejecutar y poblar la base de datos mongo. Como se ha explicado anteriormente, la base de datos se ejecuta en un contenedor e inicialmente estará vacío. Como primer paso es necesario procesar los ficheros JSON almacenados en la carpeta 'courses', que será descrita más adelante, y obtendremos un fichero 'data.json'. Con este fichero levantaremos otro servicio encargado de poblar la base de datos.

- Dockerfile: fichero en el que se recoge la información necesaria para la creación del contenedor. Se trata de una imagen de mongo 3.2.
- mongo_data_seed: se trata de un directorio que contiene los ficheros necesarios para la creación de un servicio que se encargue de la población de la base de datos:
  - precess-course.py: este fichero se encarga de la lectura de los ficheros json de todos los cursos y los procesa para que tenga un formato válido para ser importado en la base de datos.
  - data.json: fichero de salida del script anterior.
  - Dockerfile: fichero en el que se recoge la información necesaria para la creación del contenedor. Se trata de una imagen de mongo 3.2 la cual crea una conexión con la base de datos mongo para importar el fichero json procesado.

- motor: En este directorio, además de los scripts que pasaremos a explicar a continuación, usaremos dos directorios adicionales para el almacenamiento de datos. El primero de ellos, courses, contiene todos los ficheros sin procesar de los cursos. El segundo directorio, models, almacena los modelos de la representación vectorial de las palabras generados tras el procesamiento de los ficheros. La estructura del directorio 'courses' es la siguiente:

  - Nombre del curso: existe un directorio por cada uno de los cursos que analizará la aplicación, por ejemplo, UAMx+Equidad801x+3T2016. Dentro de cada uno de estos directorios, tenemos la siguiente jerarquía de carpetas:
    - certificate: contiene un fichero csv con la calificación y el tipo de certificado elegido en el curso en cuestión.
    - course: contiene dos ficheros que proporcionan información acerca de los usuarios inscritos en el curso y su estructura. Se proporcionan dos ficheros, uno en formato JSON, que provén la información estructural del curso, y otro fichero SQL con los usuarios registrados en el curso.
    - profile: contiene un fichero csv que contiene información de los usuarios inscritos.
    - social: contiene un fichero csv que recoge las interacciones de los usuarios en los foros de debate.
    - static: contiene todo el contenido estático del curso, es decir, las preguntas y los componentes de texto del curso. En el directorio 'preg' se almacenan una serie de carpetas con el número de la semana al que pertenece la pregunta, y cada una de ellas se escribe en un fichero con un formato propio denominado '.preg', donde la primera línea será el enunciado de la pregunta, y las líneas posteriores serán las posibles respuestas. Por otro lado, el directorio 'text' contendrá todos los documentos PDF y componentes HTML de cada curso.

  Este módulo se encarga del procesamiento y manipulación de todos los datos almacenados y de la generación de los indicadores que serán mostrados en la interfaz gráfica. Los scripts de los que se compone son los siguientes:

- app_motor.py: fichero que implementa la clase App para la inicialización de las bases de datos y del motor.
- config.py: fichero con las configuraciones necesarias para crear las conexiones necesarias con las bases de datos. Contiene una clase Config de la cuál heredan las clases ProductionConfig, DevelopmentConfig y TestingConfig. La instanciación de una u otra clase depende del entorno en el que se esté trabajando. Para el caso del proyecto se ha instanciado la clase DevelopmentConfig.
- data_download.py y data_download_utils.py: estos ficheros implementan la clase DataDownload y se encarga de, a partir de los enlaces a los vídeos almacenados en la plataforma de YouTube, crear la conexión necesaria para la descarga de los subtítulos de estos vídeos. Una vez descargados, se procesarán, se generarán los indicadores y se almacenarán en la base de datos.
- data_import.py: implementación de la clase DataImport, encargada de crear la conexión con la base de datos MySQL especificando el nombre de la base de datos y la codificación en la que se está trabajando (en este caso se trata de la base de datos 'datacourses' y la codificación es 'utf8mb4').
- data_upload.py: implementación de la clase DataUpload que hereda de DataImport. Se encarga de instanciar todas las clases necesarias para la carga de datos.
- data_upload_elem.py: clase abstracta que contiene la información referente a la conexión con la base de datos y las rutas a los directorios que contienen los ficheros a importar.
- data_upload_certificates.py: implementación de la clase UploadCertificates que hereda de UploadElem y que se encarga de la importación de los certificados.
- data_upload_courses.py: implementación de la clase UploadCourses que hereda de UploadElem y que se encarga de la importación de los cursos.
- data_upload_profiles.py: implementación de la clase UploadProfiles que hereda de UploadElem y que se encarga de la importación de los usuarios.
- data_upload_questions.py: implementación de la clase UploadQuestions que hereda de UploadElem y que se encarga de la importación de las preguntas y sus correspondientes respuestas.
- data_upload_social.py: implementación de la clase UploadCertificates que hereda de UploadElem y que se encarga de la importación de los certificados.
- data_upload_text.py y data_upload_text_utils.py: implementación de la clase UploadText que hereda de UploadElem y que se encarga de la importación de todos los textos de los cursos.
- data_upload_user_course.py: implementación de la clase UploadText que hereda de UploadElem y que se encarga de la importación de los datos que relaciona a los usuarios, con los cursos a los que pertenecen.
- db_utils.py: clase para facilitar la interacción con la base de datos. Permite llevar a cabo tareas como la inserción de elementos o la generación de consultas.

- index_motor.py: fichero de inicio del sistema. Instancia una clase App, asigna las configuraciones e inicia el servicio.
- model_mongo.py: clase que crea un modelo de la base de datos mongo. Recoge la información general de las distintas secciones del curso, especificando su categoría, su identificador, el identificador de las secciones hija de la actual y la metainformación asociada a dicha sección.
- model_sql.py: contiene las clases para la creación de los modelos de las distintas tablas de la base de datos SQL.
- motor.py: implementación de la clase Motor la cual se encarga de iniciar el proceso de importación y descarga.
- parse_text.py: clase ParseText encargada de la generación de todos los indicadores.
- parse_text_html.py y parse_text_pdf: implementaciones de la clase ParseText para el procesado de los ficheros en función del tipo de texto del que se trate.
- requirements.txt: fichero en el que se especifican las librerías de python necesarias para la ejecución del programa.
- sentiment_analysis.py: implementación de la clase SentimentAnalysis encargada de calcular los indicadores correspondientes a los sentimientos.
- vectorize_words.py: implementación de la clase VectorizeWords encarga de generar una representación vectorial de las palabras.

- sql:
  - Dockerfile: fichero en el que se recoge la información necesaria para la creación del contenedor. Se trata de una imagen de MySQL 8.0.3.
  - sql_seed: se trata de un directorio que contiene los ficheros necesarios para la creación de un servicio que se encargue de la población de la base de datos.
    - db.sql: fichero que contiene la estructura y las relaciones de la base de datos relacional.
    - Dockerfile: fichero en el que se recoge la información necesaria para la creación del contenedor. Se trata de una imagen de MySQL 8.0.3 la cual crea una conexión con la base de datos sql para importar el fichero db.sql.
- view: recoje todos los ficheros necesarios para la implementación un servicio que provea de una interfaz web a la aplicación.
  - static: este directorio contiene toda la información estática. Dentro de este directorio se encuentran otros cuatro directorios separados por el tipo de fichero del que se trata: css, fonts, js y templates.
  - app.py: fichero que implementa la clase App para la inicialización de las bases de datos y la definición de las rutas de las distintas vistas de la que se compone la aplicación.

- config.py: fichero con las configuraciones necesarias para crear las conexiones necesarias con las bases de datos. Contiene una clase Config de la cuál heredan las clases ProductionConfig, DevelopmentConfig y TestingConfig. La instanciación de una u otra clase depende del entorno en el que se esté trabajando. Para el caso del proyecto es DevelopmentConfig.
- controller_course_data.py: controlador para la navegación entre las secciones del curso.
- controller_courses_general.py y controller_courses_general_text.py: controladores encargados de las peticiones de los indicadores generales de los archivos de texto. Con la respuesta obtenida generamos las gráficas oportunas y renderizamos las plantillas dashboard_courses_general.html la cual se servirá de dashboard_courses_general_pdf.html y de dashboard_courses_general_video.html.
- controller_courses_social.py: controlador para crear la petición y procesar la información de las interacciones de los usuarios en el foro. Una vez generados los gráficos renderizará la plantilla dashboard_general_social.html.
- controller_courses_users.py: controlador de los componentes para la generación de gráficos referentes a la información general de los usuarios. La información la mostrará sobre la plantilla dashboard_courses_social.html.
- controller_general.py, controller_general_text.py y controller_general_video.py: estos tres ficheros son los encargados de la generación de informes sobre los ficheros de texto. Por un lado, los ficheros controller_general_text y controller_general_video se encargarán de realizar las peticiones a la API oportunas. Una vez recogidas las respuestas, el fichero controller_general se encargará de la generación de los gráficos y los mostrará en pantalla renderizando las plantillas dashboard_general_pdf.html y dashboard_general_video.html.
- controller_index.py: encargado de cargar la pantalla inicial de la aplicación. Para ello tendrá que obtener la estructura general del curso, almacenada en la base de datos mongo, y la información general del curso, contenida en la base de datos relacional. Todo ello se hace a través de la API y renderizará la información por medio de la plantilla index.html.
- controller_questions.py: este controlador procesa la información de la pantalla dedicada a los tests de los cursos. La información obtenida de la API se muestra sobre la plantilla dashboard_questions.html.
- controller_social.py: tras llevar a cabo las peticiones a la API para obtener la información referente a los foros, el controlador crea las gráficas y las muestra sobre la plantilla dashboard_social.html.
- controller_utils.py: fichero auxiliar para la manipulación de tipos de datos y la generación de gráficas.
- Dockerfile: fichero en el que se recoge la información necesaria para la creación del contenedor. Se trata de una imagen de Python 2.7.
- requirements.txt: fichero en el que se especifican las librerías de python necesarias para la ejecución del programa.

5.2. Implementación

Cada uno de estos directorios corresponden a un servicio con la intención de que cualquiera de ellos sea reemplazable. Siguiendo la estructura elegida, la vista no tiene interacción directa a las bases de datos, ya que esto lo hace por medio de peticiones HTTP a una API que también funciona como un servicio ajeno, por lo que ésta sería una sección fácilmente reemplazable. La declaración y el enlace entre los servicios se lleva a cabo a través del fichero docker-compose.yml situado en la raíz del proyecto. Las instrucciones que se llevan a cabo son las mostradas en las imágenes 5.1 y 5.2. Como se observa, el fichero está dividido en bloques en función del servicio y dentro de cada uno de ellos puede llevar a cabo las siguientes instrucciones:

- build: es el proceso de construir imágenes Docker usando un Dockerfile. La construcción usa un Dockerfile y un 'contexto'. El contexto es el conjunto de archivos en el directorio en el que está construida la imagen. [44]
- command: comando que se va a ejecutar cuando se levante el servicio.
- ports: expone los puertos del entorno virtual con el host.
- volumes: monta rutas de host o volúmenes. [44]
- links: enlaza el contenedor con otros servicios. [44]
- environment: añade variables de entorno.

Si se desease ampliar la aplicación añadiendo un nuevo servicio sería necesario crear un nuevo bloque para la definición de sus propiedades. En primer lugar, se define su etiqueta, la cual permitirá vincular el servicio con otros. En el comando build se especifica la ruta en la que se encuentra el documento Dockerfile con la configuración del Docker. Generalmente querremos que nuestros servicios se comuniquen, por lo que será necesario configurar la exposición de los puertos. Dentro del entorno virtual que se crea y finalmente se establecerán los servicios con los que el servicio en cuestión tiene que comunicarse.

Es necesario mantener la modularidad y no asignar de carga de trabajo a los servicios que no le corresponden. Por tanto, si se añade un nuevo indicador a la aplicación tendremos que proporcionar de medios de comunicación a la interfaz gráfica con las bases de datos. Esto lo haremos asignando una nueva URI a la API REST y, de esta forma, obtener la nueva información generada por medio de peticiones HTTP.

```
view:
    build: ./view
    command: python -u view/index_view.py
    ports:
      - "5000:5000"
    volumes:
      - .:/todo
    links:
      - api

motor:
    build: ./motor
    command: python -u motor/index_motor.py
    ports:
      - "6000:6000"
    volumes:
      - .:/todo
    links:
      - mysql
      - mongodb

api:
    build: ./api
    command: python -u api/index_api.py
    ports:
      - "8000:8000"
    volumes:
      - .:/todo
    links:
      - mysql
      - mongodb
```

Figura 5.1: Fichero docker-compose.yml -1-


```
view:
    build: ./view
    command: python -u view/index_view.py
    ports:
      - "5000:5000"
    volumes:
      - .:/todo
    links:
      - api

motor:
    build: ./motor
    command: python -u motor/index_motor.py
    ports:
      - "6000:6000"
    volumes:
      - .:/todo
    links:
      - mysql
      - mongodb

api:
    build: ./api
    command: python -u api/index_api.py
    ports:
      - "8000:8000"
    volumes:
      - .:/todo
    links:
      - mysql
      - mongodb
```


Figura 5.2: Fichero docker-compose.yml -2-

6 Pruebas

A lo largo del capítulo se describirán las pruebas que se han llevado a cabo para comprobar el correcto funcionamiento y la completitud del sistema.

6.1. Pruebas unitarias

En esta sección, se explicarán las pruebas que se han llevado a cabo para verificar la funcionalidad de cada módulo.

- Para el Módulo de almacenamiento, se ha comprobado la correcta gestión y conexión de las bases de datos haciendo uso de un contenedor Docker auxiliar para realizar una simulación de consultas. Además, se ha comprobado el funcionamiento de los contenedores encargados de la importación de copias de seguridad a las bases de datos usando imágenes en diferentes estados.

- Para el Módulo de Cálculo, se ha comprobado la información que generaba y la conexión con el Módulo de almacenamiento comprobando que la salida de los procesos que realizaba el módulo coincidía con el contenido en la base de datos.

- Para la API se han llevado a cabo pruebas basadas en peticiones HTTP a través del navegador Web logrando ver que la salida visualizada es la esperada.

- Para el Módulo de Interfaz gráfica se han hecho pruebas basadas en el correcto funcionamiento de la interfaz web. Se ha comprobado que los controladores generen la respuesta correcta y que la información mostrada es la esperada.

6.2. Pruebas de integración

En esta sección, se explicarán las pruebas realizadas para comprobar si el flujo de información entre las distintas capas es el adecuado.

- Por un lado, se ha probado el Módulo de cálculo junto con el Módulo de almacenamiento, comprobando que los resultados que se obtienen del cálculo coinciden con el que se encuentra en la base de datos.
- Por otro lado, se ha comprobado la API con el Módulo de almacenamiento, generando una serie de peticiones y comprobando la salida obtenida.
- Finalmente, se ha comprobado la Interfaz gráfica con la API, comprobando que los controladores generasen adecuadamente las peticiones y que las gráficas representen los datos adecuados.

6.3. Pruebas de sistema

En esta sección, se explicarán las pruebas realizadas para comprobar que la aplicación funciona en distintos entornos. Para ello, se ha comprobado la aplicación en un ordenador MacBook Pro con SSD y en una Raspberry PI2. El resultado obtenido es que, mientras el procesamiento de cálculo es bueno para el MacBook, la ejecución sobrecarga la Raspberry. No obstante, con los datos ya precalculados, la ejecución de la API y de la Interfaz gráfica es adecuada en ambos dispositivos.

6.4. Pruebas de validación

En esta sección, se han llevado a cabo las pruebas de validación al contrastar que se han cumplido todos los requisitos funcionales definidos.

6.5. Pruebas de aceptación

En esta sección, se han llevado a cabo una serie de pruebas para comprobar que el producto realizado concuerda con el deseado. Para ello, se han definido una serie de casos de prueba definidos en el anexo B donde un participante, sin familiaridad con el uso de la aplicación, ha comprobado la usabilidad, accesibilidad y funcionalidad de la aplicación.

7

Conclusiones y trabajo futuro

Finalmente, se expondrán las conclusiones más importantes que se han obtenido durante el proceso de elaboración del proyecto, así como los posibles trabajos futuros que se podrán realizar teniendo como base este proyecto.

7.1. Conclusiones

A lo largo de este TFG se ha diseñado y desarrollado un sistema software que tiene como objetivo ser una herramienta que facilite la aplicación de técnicas de Procesamiento del Lenguaje Natural sobre el contenido de cursos MOOC, llevando consigo una serie de procesos de carga, preprocesamiento, limpieza y transformación de los datos.

El proyecto ha sido tomado como finalizado cuando se han desarrollado los objetivos fijados en el apartado 1.2 y que se describen a continuación.

Se ha llevado a cabo un estudio sobre los datos de los MOOC proporcionados por la plataforma edX y se han procesado para que puedan ser tratados por el sistema.

Se ha llevado a cabo un estudio sobre las diferentes técnicas del Procesamiento del Lenguaje Natural y de la arquitectura basada en microservicios.

Se ha diseñado un sistema modular y fácilmente escalable que permite:

- Cargar y procesar cursos MOOC.
- Acceder a información general sobre el trascurso de los cursos MOOC.
- Visualizar gráficas e indicadores con información relativa a los componentes de texto.

- Exportar los datos utilizados para la generación de las gráficas.

Gracias al uso de microservicios, ha sido posible crear una aplicación que podrá ser ampliada con distintos focos de estudio. Con estas técnicas de virtualización se hace posible que la aplicación crezca de forma simple y organizada facilitando el reparto de tareas y teniendo que dedicar menos tiempo y esfuerzo en la integración de los nuevos focos de estudio. Esta organización también permite usar los servicios realizados en esta aplicación con otros objetivos, haciéndolo fácil de integrar en otras programas.

El uso de diferentes tecnologías ha permitido ampliar mis conocimientos sobre distintas áreas de la Informática. Se ha hecho uso de Sistemas de Gestión de Bases de Datos tanto relacionales como no relacionales, permitiendo conocer y manejar los dos principales tipos de bases de datos utilizadas en la actualidad. Debido al uso de microservicios, se han desarrollado métodos de comunicación entre servicios familiarizándome con el uso, la implementación y el funcionamiento de las APIs. Además, se ha hecho uso de una amplia variedad de librerías de código abierto de Python, permitiendo conocer y modificar estas librerías para adaptarlas al objetivo del TFG. Con todo esto, se ha creado una aplicación que une una amplia variedad de tecnologías que, a pesar del crecimiento, es fácilmente escalable y modificable hacia distintos fines.

7.2. Trabajo Futuro

Uno de los aspectos más destacables de la aplicación es la modularidad, lo cual facilitará el proceso de ampliación del programa hacia nuevos focos de análisis.

Por un lado, se propone un análisis sobre el contenido visual de los componentes de video, siendo interesante recoger elementos como la iluminación, gama de colores o velocidad de transición.

Sería interesante llevar a cabo un análisis y una generación de modelos para realizar un análisis predictivo para detectar riesgos de abandono o probabilidad de terminar el curso satisfactoriamente.

El objetivo de esta herramienta es poder llegar a ser un programa completo para el análisis de contenido audiovisual.

Bibliografía

[1] Gayle Christensen y col. «The MOOC phenomenon: who takes massive open online courses and why?» En: *ssrn* (2013).

[2] UAM. *Universidad Autónoma de Madrid*. URL: `http://www.uam.es/UAM/Home.htm?language=es` (visitado 14-06-2018).

[3] Ruth Cobos y col. «Towards a collaborative pedagogical model in MOOCs». En: *EDUCON* (2014), pág. 83.

[4] Rodrigo Saraguro Bravo y col. «USO DE TÉCNICAS DE GAMIFICACIÓN EN EL DISEÑO TECNOPEDAGÓGICO DE UN MOOC». En: *Libro de Comunicaciones* (2016), pág. 83.

[5] Mary Dee Harris. *Introduction to natural language processing*. Reston Publishing Co., 1985.

[6] Ruth Cobos y col. «Proyecto eMadrid: MOOCs y Analítica del Aprendizaje». En: *XVIII SIMPOSIO INTERNACIONAL DE INFORMÁTICA EDUCATIVA, SIIE 2016*. 2016, pág. 491.

[7] Armin Ronacher. *Flask*. URL: `https://github.com/pallets/flask` (visitado 14-06-2018).

[8] Coursera. *Coursera*. URL: `https://www.coursera.org/` (visitado 14-06-2018).

[9] FutureLearn. *FutureLearn*. URL: `https://www.futurelearn.com/` (visitado 14-06-2018).

[10] J Brookshear. *Glenn: Teoría de la Computación. Lenguajes formales, autómatas y complejidad*. 1993.

[11] Augusto Cortez Vásquez, Jaime Pariona Quispe, Ana Maria Huayna y col. «Procesamiento de lenguaje natural». En: *Revista de investigación de Sistemas e Informática* 6.2 (2009), págs. 45-54.

[12] Manuel Montes y Gómez, Alexander Gelbukh y Aurelio López López. «Minería de texto empleando la semejanza entre estructuras semánticas». En: *Computación y Sistemas* 9.1 (2005), págs. 63-81.

[13] Lina Andrea Torres Samboni. «Análisis de sentimientos sobre el posconflicto colombiano utilizando herramientas de minería de texto». En: *Análisis de sentimientos sobre el posconflicto colombiano utilizando herramientas de minería de texto* (2016).

[14] Luis Armando Llumiquinga. «Desarrollo de una metodología para el reconocimiento de los niveles de estrés». En: (2017).

[15] E. Agirre y col. «XUXEN: A Spelling Checker/Corrector for Basque Based on Two-level Morphology». En: *Proceedings of the Third Conference on Applied Natural Language Processing*. Ed. por Association for Computational Linguistics. ANLC '92. Trento, Italy: Association for Computational Linguistics, 1992, págs. 119-125. DOI: 10.3115/974499.974520. URL: https://doi.org/10.3115/974499.974520.

[16] Anjali Ganesh Jivani y col. «A comparative study of stemming algorithms». En: *Int. J. Comp. Tech. Appl* 2.6 (2011), págs. 1930-1938.

[17] Zellig S Harris. «Distributional structure». En: *Word* 10.2-3 (1954), págs. 146-162.

[18] Tomas Mikolov y col. «Efficient estimation of word representations in vector space». En: *arXiv preprint arXiv:1301.3781* (2013).

[19] Petr Sojka y Radim Hatlapatka. «Document Engineering for a Digital Library: PDF Recompression Using JBIG2 and Other Optimizations of PDF Documents». En: *Proceedings of the 10th ACM Symposium on Document Engineering*. DocEng '10. Manchester, United Kingdom: ACM, 2010, págs. 3-12. DOI: 10.1145/1860559.1860563. URL: http://doi.acm.org/10.1145/1860559.1860563.

[20] Spark MLlib. *Word2vec implementation in Spark MLlib*. URL: https://spark.apache.org/docs/latest/mllib-feature-extraction.html (visitado 14-06-2018).

[21] ITS. *Office of Information Technology Service*. URL: http://www.its.state.nc.us/Information/Glossary/Glossm.asp (visitado 14-06-2018).

[22] Chris Richardson. *Microservice Architecture*. URL: http://microservices.io/patterns/monolithic.html (visitado 14-06-2018).

[23] Sam Newman. *Building microservices*. Ed. por Beijing. O'Reilly, 2015. ISBN: 978-1-491-95035-7.

[24] Martin Fowler. *Microservices*. URL: https://www.martinfowler.com/articles/microservices.html (visitado 14-06-2018).

[25] Mark Richards. *Software architecture patterns : understanding common architecture patterns and when to use them*. Ed. por Sebastopol. O'Reilly Media, 2015. ISBN: 978-1-491-92424-2.

[26] Roy T. Fielding y Richard N. Taylor. «Principled Design of the Modern Web Architecture». En: *Proceedings of the 22Nd International Conference on Software Engineering*. ICSE '00. Limerick, Ireland: ACM, 2000, págs. 407-416. ISBN: 1-58113-206-9. DOI: 10.1145/337180.337228. URL: http://doi.acm.org/10.1145/337180.337228.

[27] materializecss. *materializecss*. URL: https://materializecss.com/ (visitado 14-06-2018).

[28] Francisco Javier Moreno, Juan Esteban Quintero y Robinson Rueda. *UNA COMPARACIÓN DE RENDIMIENTO ENTRE ORACLE Y MONGODB*. URL: http://www.redalyc.org/html/911/91145342001/ (visitado 14-06-2018).

[29] MongoDB. *MongoDB*. URL: https://www.mongodb.com/collateral/mongodb-architecture-guide (visitado 14-06-2018).

[30] Ewan Klein Steven Bird y Edward Loper. *Natural Language Processing with Python*. O Reilly, 2009.

[31] EAGLES. *INTRODUCCIÓN A LAS ETIQUETAS EAGLES*. URL: http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html/ (visitado 14-06-2018).

[32] Gensim. *Gensim. Topic modelling*. URL: https://radimrehurek.com/gensim/ (visitado 14-06-2018).

[33] Word2Vec. *Word2Vec*. URL: https://code.google.com/archive/p/word2vec/ (visitado 14-06-2018).

[34] Steven Loria. *TextBlob*. URL: https://github.com/sloria/TextBlob (visitado 14-06-2018).

[35] C. J. Hutto y E. Gilbert. «Vader: A parsimonious rule-based model for sentiment analysis of social media text». En: *In 8th International AAAI Conference on Weblogs and Social Media (ICWSM)* (2014).

[36] Matthew Stamy. *PyPDF2*. URL: https://github.com/mstamy2/PyPDF2 (visitado 14-06-2018).

[37] Andreas Mueller. *Wordcloud*. URL: http://amueller.github.io/word_cloud/ (visitado 14-06-2018).

[38] Leonard Richardson. *Beautiful Soup*. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (visitado 14-06-2018).

[39] Community. *Pandas*. URL: https://pandas.pydata.org/ (visitado 14-06-2018).

[40] Florian Mounier. *PyGal*. URL: http://pygal.org/en/stable/ (visitado 14-06-2018).

[41] Ricardo Garcia. *youtube-dl*. URL: https://github.com/rg3/youtube-dl (visitado 14-06-2018).

[42] Fabian Pedregosa y col. «Scikit-learn: Machine Learning in Python». En: *Journal of Machine Learning Research 12* (2012).

[43] edX. *edX*. URL: https://www.edx.org/es (visitado 14-06-2018).

[44] Docker. *Compose*. URL: https://docs.docker.com/compose/compose-file/ (visitado 14-06-2018).

Apéndices

A

Manual de usuario de la herramienta

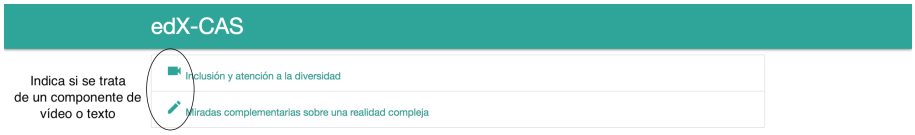

Figura A.1: Pantalla para seleccionar el componente a visualizar

Figura A.2: Pantalla principal

Figura A.3: Pantalla de información sobre los usuarios

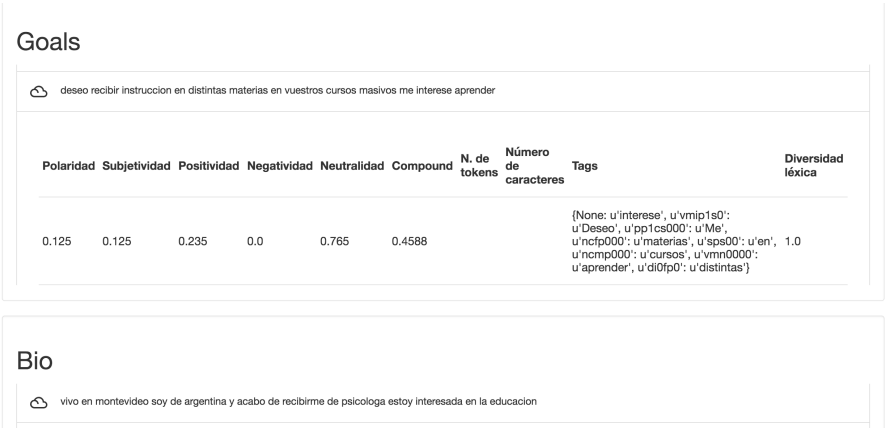

Figura A.4: Tabla con información de la descripción de los usuarios

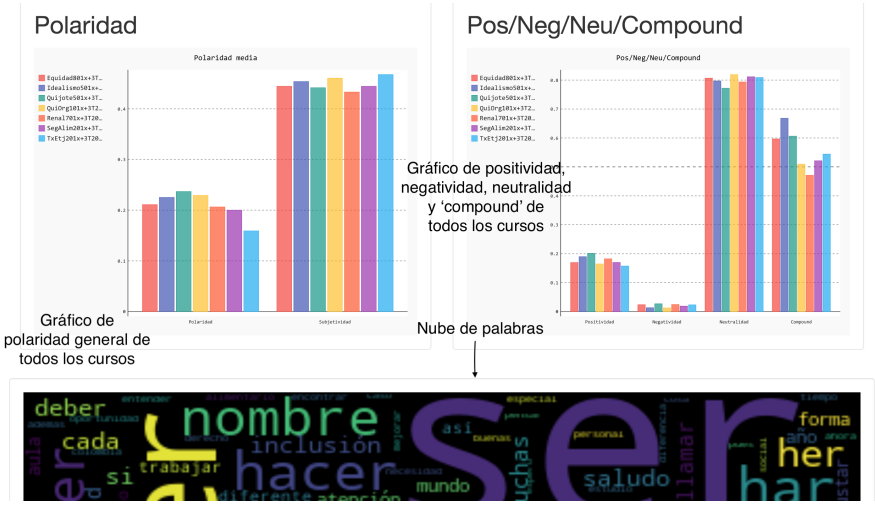

Figura A.5: Tabla con gráficos sobre los usuarios

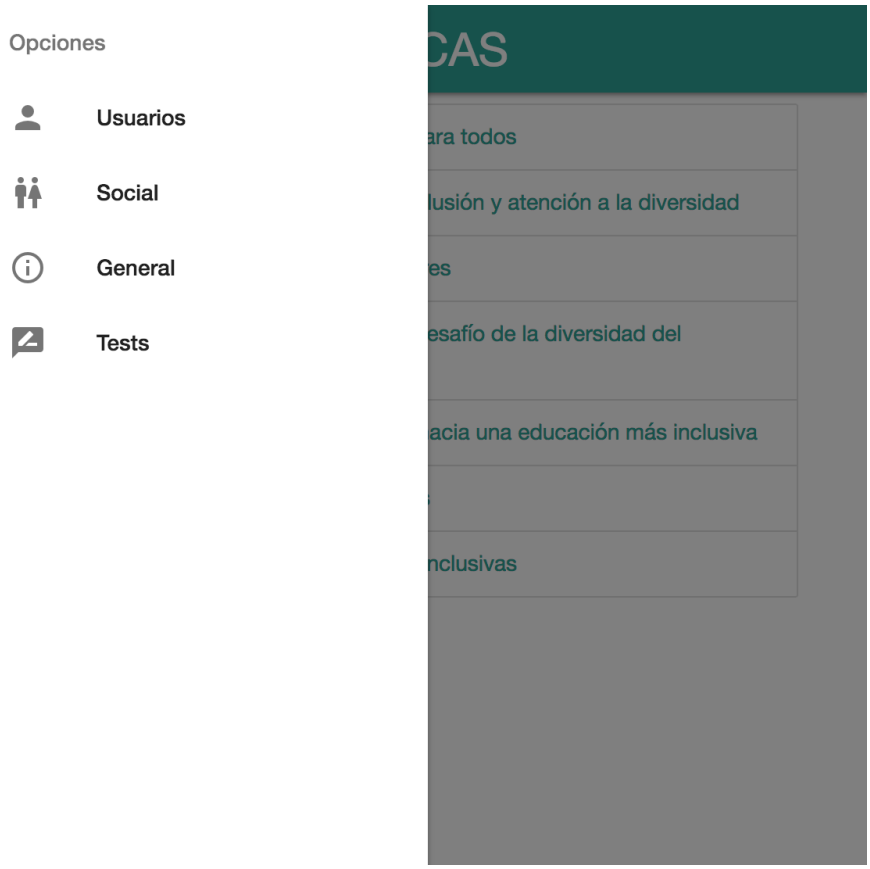

Figura A.6: Índice del curso en formato móbil

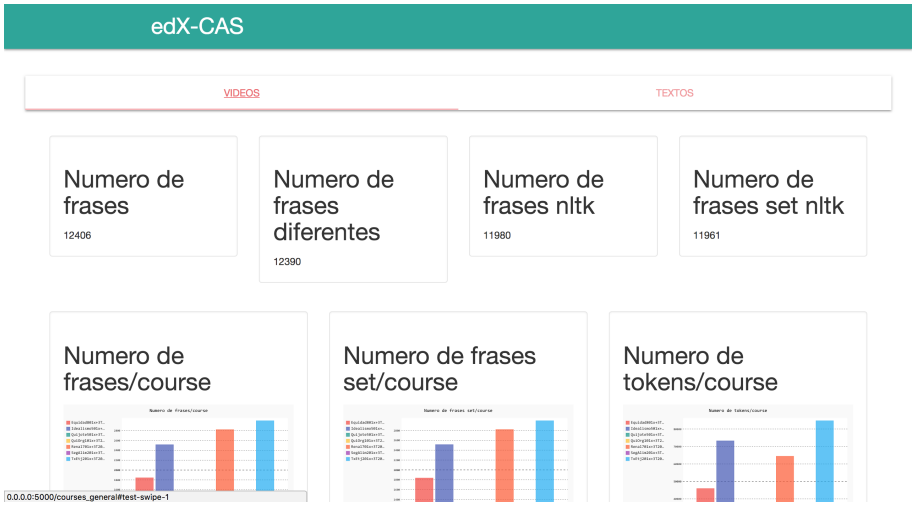

Figura A.7: Dashboard de información general de los textos de los cursos

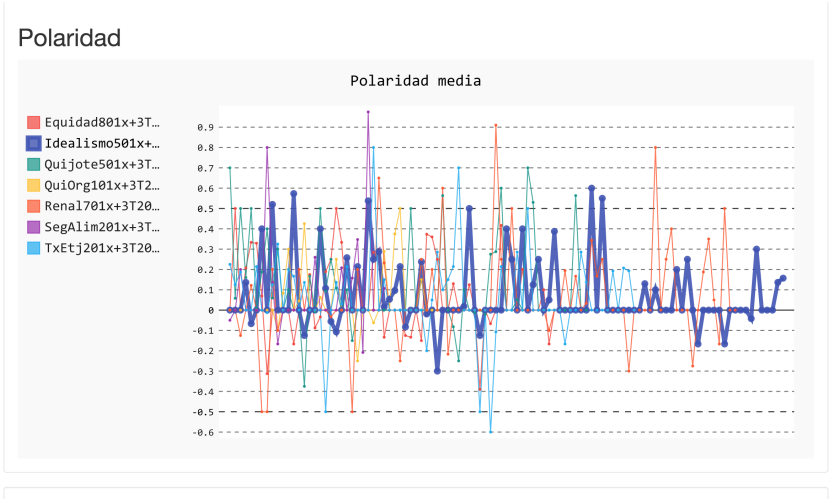

Figura A.8: Índice de los componentes de texto

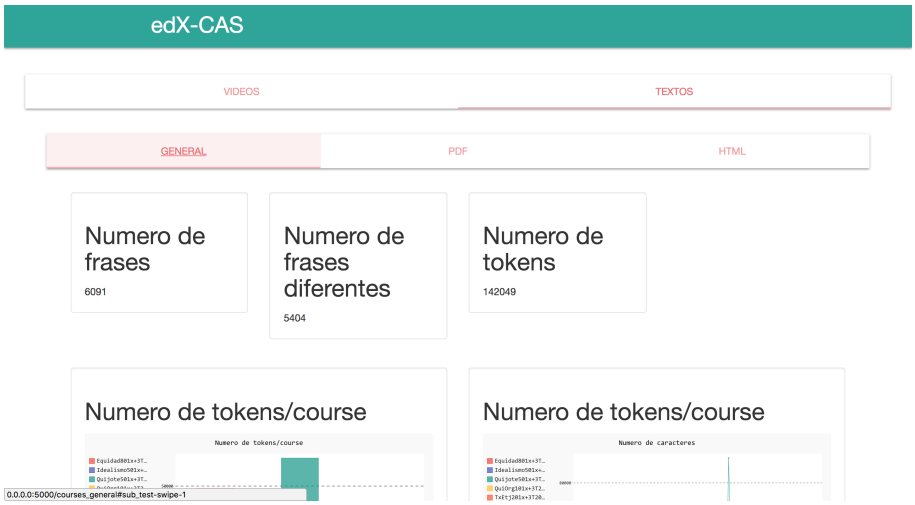

Figura A.9: Logo de la Universidad Autónoma de madrid.

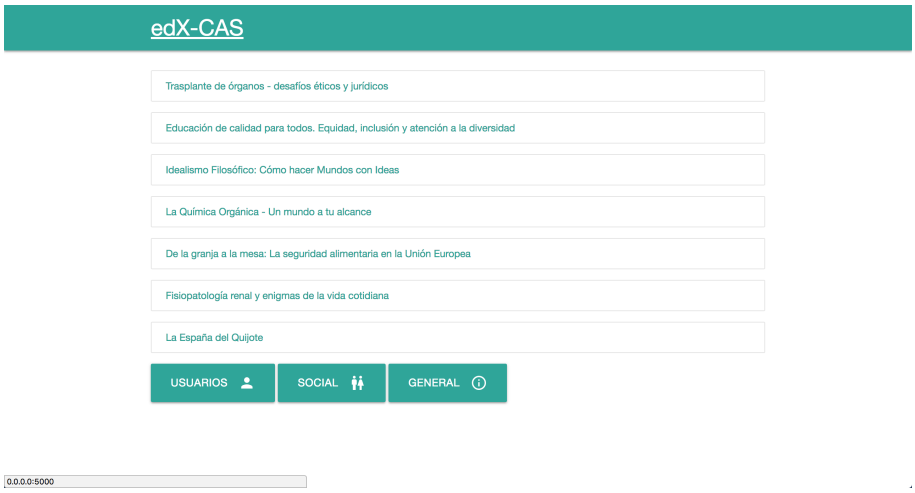

Figura A.10: Índice de la aplicación

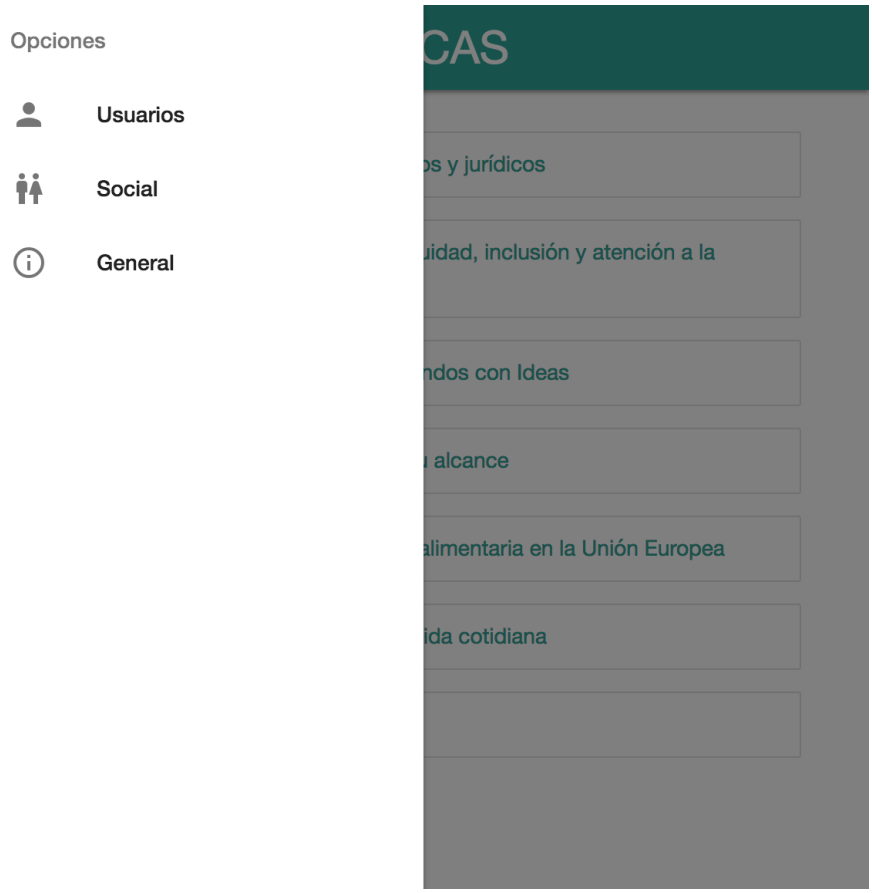

Figura A.11: Índice de la aplicación en formato móbil

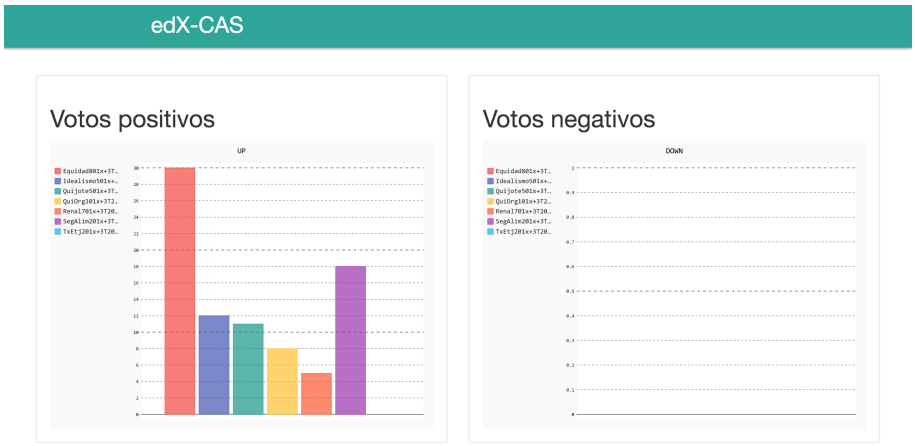

Figura A.12: Pantalla de información de los foros

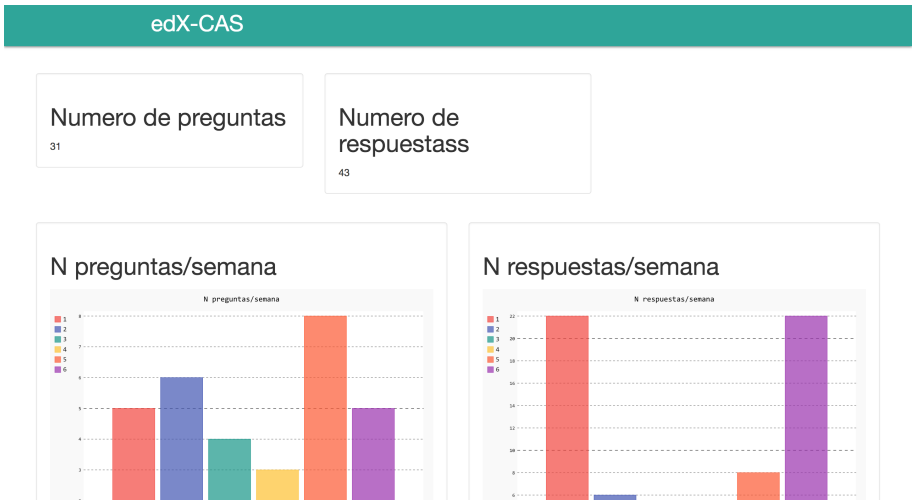

Figura A.13: Pantalla de información a las preguntas

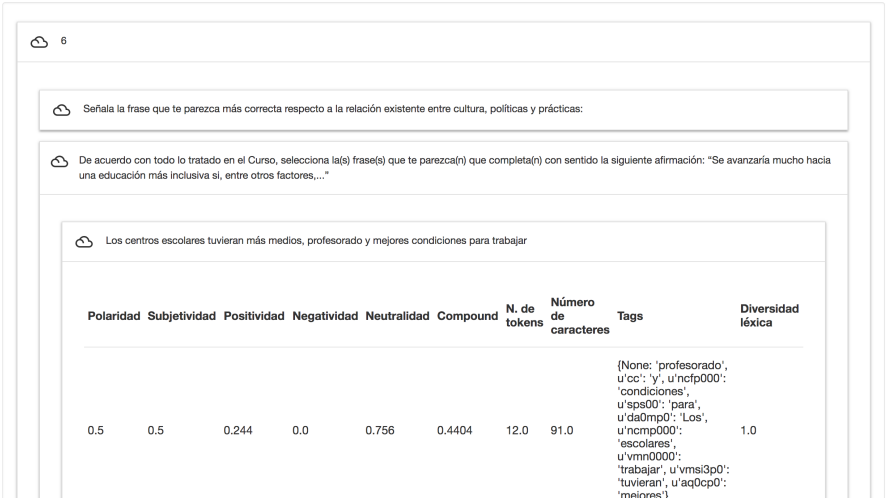

Figura A.14: Pantalla de información a las preguntas

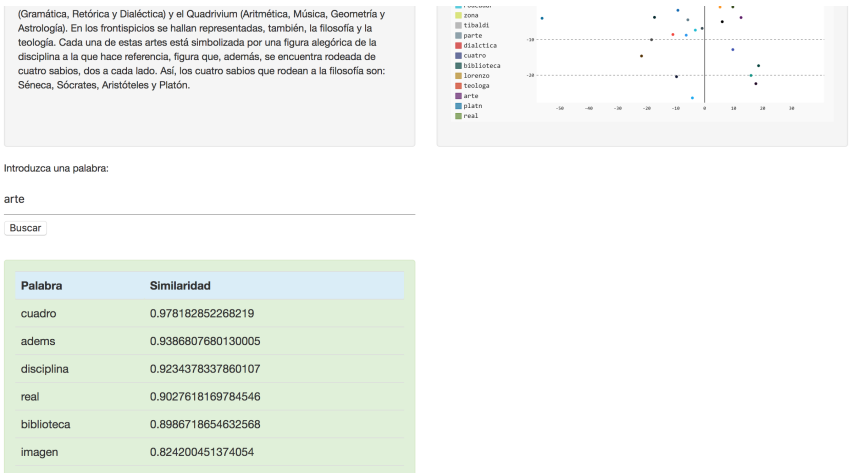

Figura A.15: Similutud en componentes de texto

Figura A.16: Nube de palabras en textos

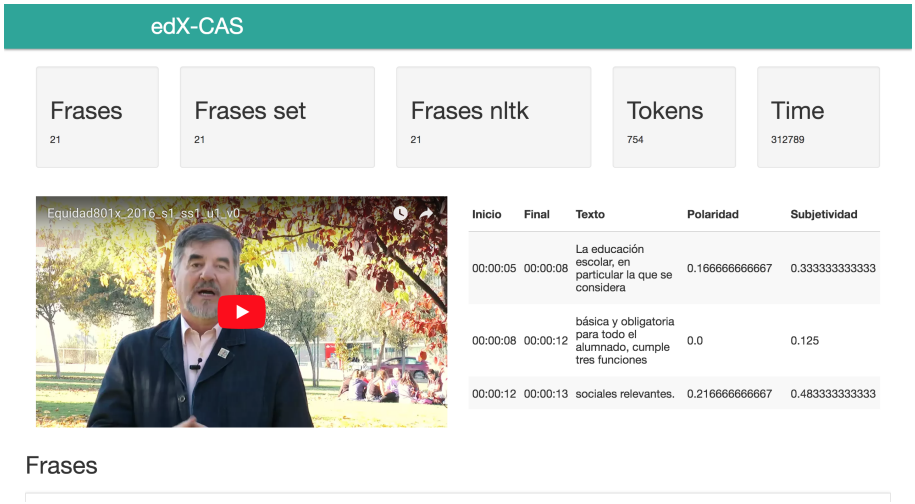

Figura A.17: Pantalla de componente de vídeo

Figura A.18: Pantalla con información sobre las frases de un componente de vídeo

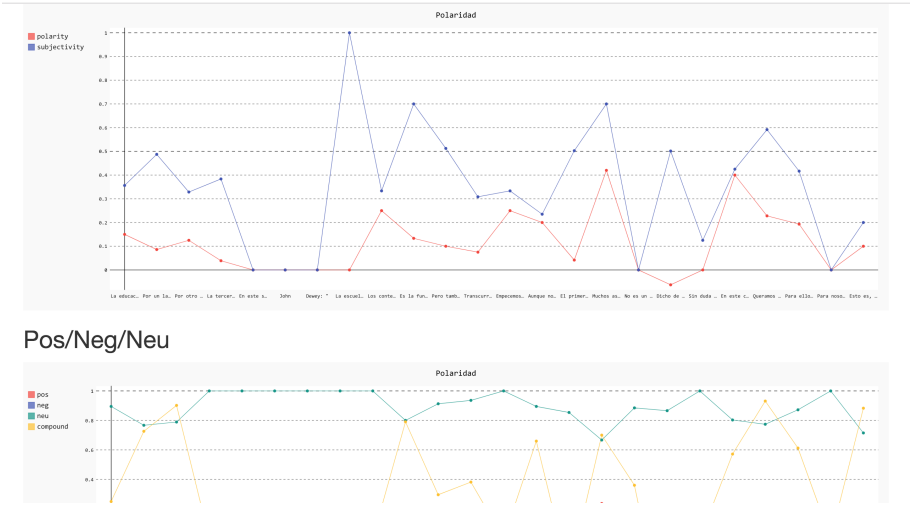

Figura A.19: Gráficos de un componente

B

Planificación

En este anexo se muestra el diagrama de Gantt B.1 seguido para completar la implementación del software. Se pueden diferenciar seis etapas principales, todas ellas realizadas de manera secuencial hasta el punto de la implementación, donde se ha podido paralelizar parcialmente el módulo de cálculo y el de la visualización.

Etapas proyecto	Recursos	Estado
Propuesta	**Nombre**	**Terminado**
Propuesta	Álvaro Villén	Sí
Definición	**Nombre**	**Terminado**
Primera organización de análisis	Álvaro Villén	Sí
Definición del problema	Álvaro Villén	Sí
Concepto	**Nombre**	**Terminado**
Diseño web	Álvaro Villén	Sí
Descripción de concepto	Álvaro Villén	Sí
Diseño	**Nombre**	**Terminado**
Arquitectura lógica	Álvaro Villén	Sí
Módulo de Almacenamiento	Álvaro Villén	Sí
Módulo de Cálculo	Álvaro Villén	Sí
Módulo de Visualización	Álvaro Villén	Sí
Implementación	**Nombre**	**Terminado**
Base de datos	Álvaro Villén	Sí
Módulo de Almacenamiento	Álvaro Villén	Sí
Módulo de Cálculo	Álvaro Villén	Sí
Módulo de Visualización	Álvaro Villén	Sí
Pruebas	Álvaro Villén	Sí
Documentación	**Nombre**	**Terminado**
Documentación	Álvaro Villén	Sí

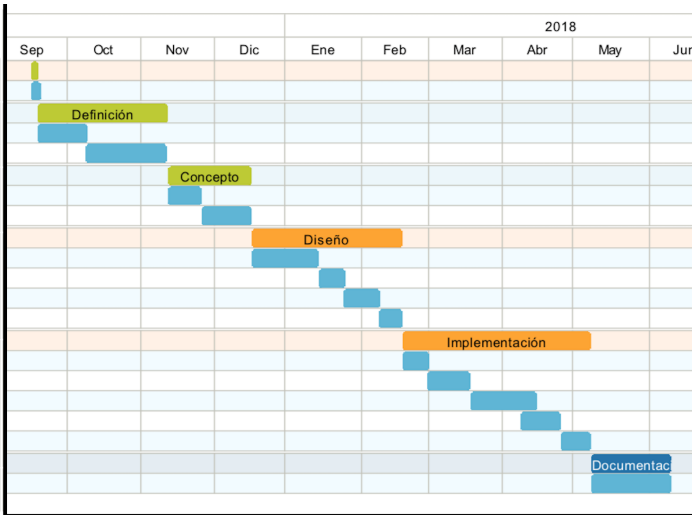

Figura B.1: Diagrama de Gantt

C

Pruebas de Aceptación

A lo largo del anexo, se presentan las pruebas que se han llevado a cabo para comprobar que el producto realizado concuerda con el deseado:

Información de los usuarios	**Req[01-01]**
Descripción: Acceso a las estadísticas de los usuarios. El usuario deberá acceder tanto a la información general de los estudiantes de todos los cursos como de cada curso en específico.	
Prerrequisitos: El usuario deberá estar ubicado en la pantalla inicial de la aplicación	
Pasos: - Sobre la pantalla principal pulsar el botón de usuario. - Posteriormente acceder a un curso y seleccionar el botón de usuario	
Resultado esperado: El usuario ha accedido satisfactoriamente a la información de usuarios de todos los cursos y de uno en específico.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.1: Prueba de aceptación: Req [01-01]

Información de los usuarios	**Req[01-01]**
Descripción: El usuario podrá analizar el contenido textual usado como descripción personal del estudiante.	
Prerrequisitos: El usuario deberá estar ubicado en la pantalla de información de usuario, siendo indiferente si se trata de un curso en específico o de todos ellos	
Pasos: - Seleccionar el botón de usuario. - Seleccionar una frase. - Observar la información que contine.	
Resultado esperado: El usuario ha accedido satisfactoriamente a la información obtenida de cada frase.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.2: Prueba de aceptación: Req [01-02]

Información de los usuarios	**Req[01-04]**
Descripción: El usuario podrá visualizar una nube de palabras con los términos más destacados de los usuarios.	
Prerrequisitos: El usuario deberá estar ubicado en la pantalla de información de usuario, siendo indiferente si se trata de un curso en específico o de todos ellos.	
Pasos: - Seleccionar el botón de usuario. - Desplazarse hasta el componente de la nube de palabras.	
Resultado esperado: El usuario ha visualizado satisfactoriamente la información obtenida de cada frase.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.3: Prueba de aceptación: Req [01-04]

Información de los foros	**Req[02-01]**
Descripción: Acceso a las estadísticas de los usuarios. El usuario deberá acceder tanto a la información general de los foros de todos los cursos como de cada curso en específico.	
Prerrequisitos: El usuario deberá estar ubicado en la pantalla inicial de la aplicación.	
Pasos: - Seleccionar el botón de usuario. - Sobre la pantalla principal pulsar el botón de social. - Posteriormente acceder a un curso y seleccionar el botón de usuario	
Resultado esperado: El usuario ha accedido satisfactoriamente a la información de los foros de todos los cursos y de uno en específico	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.4: Prueba de aceptación: Req [02-01]

Información de los foros	**Req[02-02]**
Descripción: El usuario podrá analizar el contenido textual usado de los estudiantes usados en los foros de debate.	
Prerrequisitos: El usuario deberá estar ubicado en la pantalla de información social, siendo indiferente si se trata de un curso en específico o de todos ellos.	
Pasos: - Seleccionar el botón de social. - Seleccionar una frase. - Observar la información que contiene	
Resultado esperado: El usuario ha accedido satisfactoriamente a la información obtenida de cada frase.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.5: Prueba de aceptación: Req [02-02]

Información de los foros	**Req[02-03]**
Descripción: El usuario podrá visualizar una nube de palabras con los términos más destacados de los foros de debate.	
Prerrequisitos: El usuario deberá estar ubicado en la pantalla de social, siendo indiferente si se trata de un curso en específico o de todos ellos.	
Pasos: - Seleccionar el botón de social. - Desplazarse hasta el componente de la nube de palabras	
Resultado esperado: El usuario ha visualizado satisfactoriamente la nube de palabras correspondiente a los foros de debate.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.6: Prueba de aceptación: Req [02-03]

Información de los vídeos	**Req[03-01]**
Descripción: El usuario podrá visualizar el componente de vídeo que se pretende analizar.	
Prerrequisitos: El usuario ha de desplazarse hasta una sección que contenga un componente de vídeo.	
Pasos: - Seleccionar un curso. - Seleccionar una semana. - Seleccionar una sección. - Hacer click sobre el componente de vídeo	
Resultado esperado: El usuario puede visualizar el componente de vídeo.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.7: Prueba de aceptación: Req [03-01]

Información de los vídeos	**Req[03-02]**
Descripción: El usuario podrá visualizar información estadística de los vídeos.	
Prerrequisitos: El usuario podrá visualizar información estadística de los vídeos.	
Pasos: - Seleccionar un curso. - Seleccionar un curso. - Seleccionar una semana. - Seleccionar una sección	
Resultado esperado: El usuario puede visualizar información estadística sobre los cursos.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.8: Prueba de aceptación: Req [03-02]

Información de los textos	**Req[04-01]**
Descripción: El usuario podrá visualizar el componente de texto que se pretende analizar.	
Prerrequisitos: El usuario ha de desplazarse hasta una sección que contenga un componente de texto.	
Pasos: - Seleccionar un curso. - Seleccionar una semana. - Seleccionar una sección	
Resultado esperado: El usuario puede visualizar el componente de texto.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.9: Prueba de aceptación: Req [04-01]

Información de los tests	**Req[04-02]**
Descripción: El usuario podrá visualizar las preguntas y las posibles respuestas a dichas preguntas organizados por el número de la semana en la que se ha planteado.	
Prerrequisitos: El usuario ha de situarse sobre el menú principal de la aplicación.	
Pasos: - Seleccionar el botón de preguntas.	
Resultado esperado: El usuario puede visualizar información estadística sobre el cada pregunta y cada respuesta.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.10: Prueba de aceptación: Req [04-02]

Descarga de Datos	**Req[05-01]**
Descripción: El usuario podrá descargar en ficheros csv la información almacenada en la base de datos.	
Prerrequisitos: El usuario ha de situarse sobre el menú principal de la aplicación.	
Pasos: - Seleccionar el botón de descargas.	
Resultado esperado: El usuario puede descargar información de la base de datos.	
Resultado obtenido: El resultado concuerda con el esperado.	

Tabla C.11: Prueba de aceptación: Req [05-01]